GESTION DE PROYECTOS DE SOFTWARE

Introducción intuitiva de la importancia de las técnicas de gestión y de los factores humanos

Costes del software

El factor humano en la productividad

Calidad y formación

Esfuerzo de gestión/esfuerzo técnico

El reto de la dirección de proyectos de software

Categorías de software

Tamaño del software

Software del sistema, de soporte, de aplicación

Software-proyecto y software-producto

Tipología de Brooks

Software horizontal, vertical y genérico

Software integrado

Clasificación SPE de los programas

Modelos 3P y 5P de los proyectos de software. Aproximación cibernética y ciclo de vida trifásico

Problema, producto y proceso

Experimento de Weinberg-Schulman

Tasa de cambio y borrosidad

Representación cibernética del modelo 3P

Ciclo de vida trifásico

Técnicas predominantes según las fases

Modelo 5P

Técnicas de estimación

El ciclo de estimación-planificación-control

Microestimación y macroestimación

Técnica clásica de planificación de recursos humanos. Contingencias

Concepto de productividad en software

Conflicto de cantidad y calidad

Módulos propensos a defectos

Evolución tecnológica de la productividad

Macroestimación: estudios sobre factores de productividad

Predictor de productividad de Felix/Walston/IBM

Rangos de productividad y árbol de mejora de productividad: Boehm/TRW/COCOMO

Métodos y modelos de macroestimación

Relaciones de Felix/Walston/IBM

Método de Putnam

Modelo COCOMO (TRW, Boehm)

Mantenimiento del software

Factores de coste

Categorías de mantenimiento

COCOMO, de nuevo

Relación con gestión de configuraciones y control de calidad

Factores humanos individuales

Liderazgo proyecto software

Estructura factorial de la inteligencia

Estructura factorial de la personalidad

Algo de psicología cognitiva. Aplicación en programación

Formación en ingeniería del software

Factores humanos sociales

El problema de la comunicación

Comportamiento del grupo pequeño

Equipo estructurado de programación

Leyes de los proyectos y de la naturaleza humana

Caso práctico: Metodología 4FRONT (Deloitte & Touche)

Referencias bibliográficas

Boehm, B., Software Engineering Economics, Prentice-Hall, N.J., 1981.

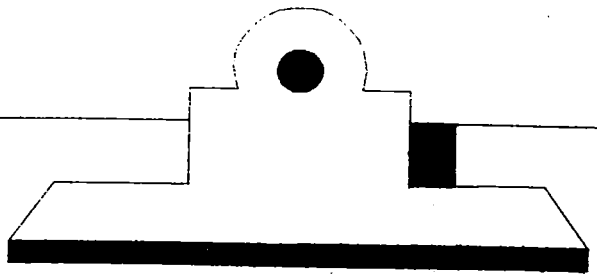
Brooks, F.P.Jr, The Mythical Man-Month. Essays on Software Engineering, Addison-Wesley, Mass., reimpresión, 1982.

Lammers, S., Programadores en Acción, Microsoft-Anaya multimedia, Madrid, 1988.

Pinillos, J.L., La Mente Humana, Ed. Temas de Hoy. Madrid, 1991.

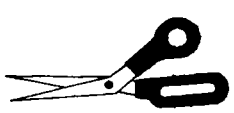
Sommerville, I., Software Engineering, Addison-Wesley, Wokingham, England, 3a ed., 1989.

Artículos seleccionados



INTRODUCCION INTUITIVA DE LA IMPORTANCIA DE LAS TECNICAS DE GESTION Y DE LOS FACTORES HUMANOS

- ☆ COSTES DEL SOFTWARE
- ☆ EL FACTOR HUMANO EN LA PRODUCTIVIDAD
- ☆ CALIDAD Y FORMACION
- ☆ ESFUERZO DE GESTION/ESFUERZO TECNICO
- ☆ EL RETO DE LA DIRECCION DE PROYECTOS DE SOFTWARE

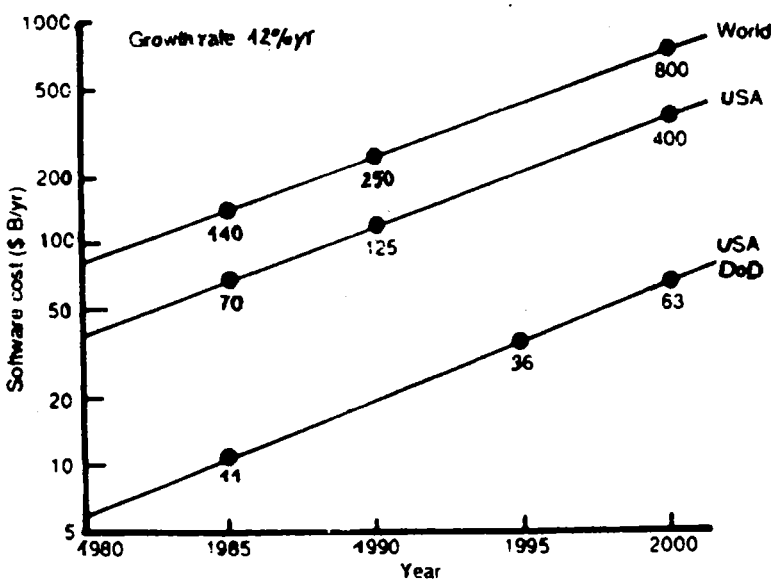


COSTES DEL SOFTWARE

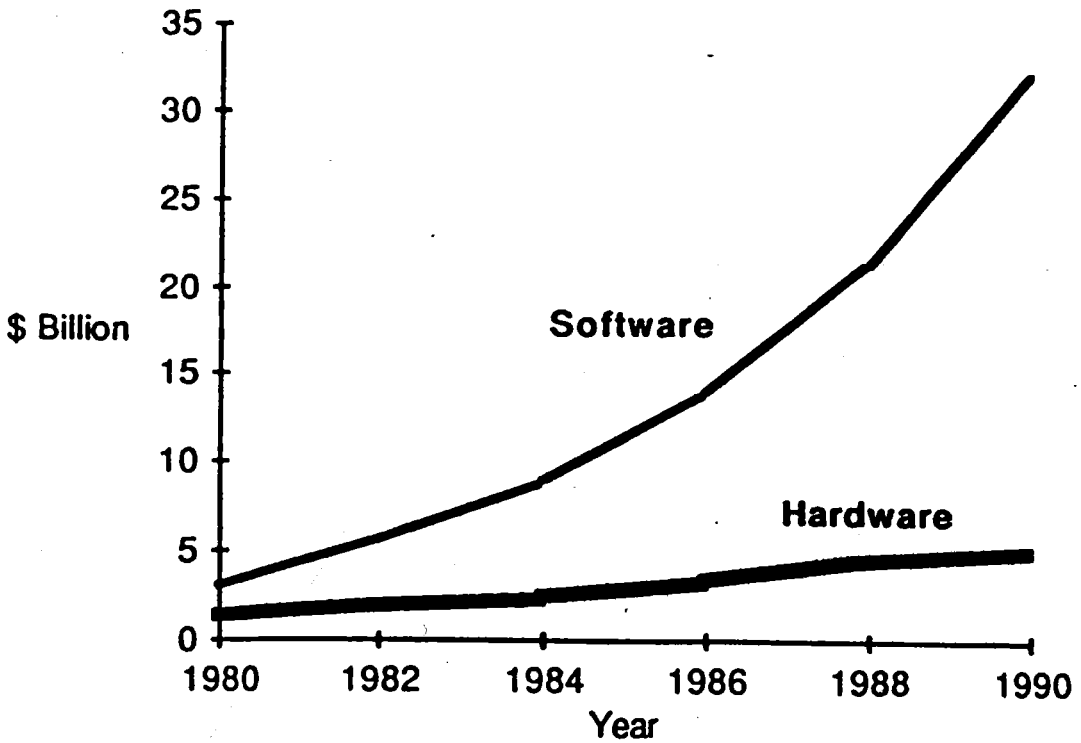
COSTE PROGRAMACION U.S.A. 1977: $\simeq$ 100.000 M$

> 3% PNB

(LEHMAN, 1980)



TENDENCIAS DE COSTES SOFTWARE (BOEHM, 1987)



Projected Growth of Software and Hardware Costs

COSTES DEL SOFTWARE

CICLO VIDA:

DESARROLLO 30 - 50%

EVOLUCION 70 - 50%

(LEHMAN, 1980)

REGLA PRACTICA DE DISTRUBUCION DEL COSTE DE DESARROLLO EN LA DECADA DE LOS 70: 40/20/40

40% ANALIZAR Y DISEÑAR

20% CODIFICAR Y DEPURAR UNIDADES

40% PROBAR Y ENTREGAR

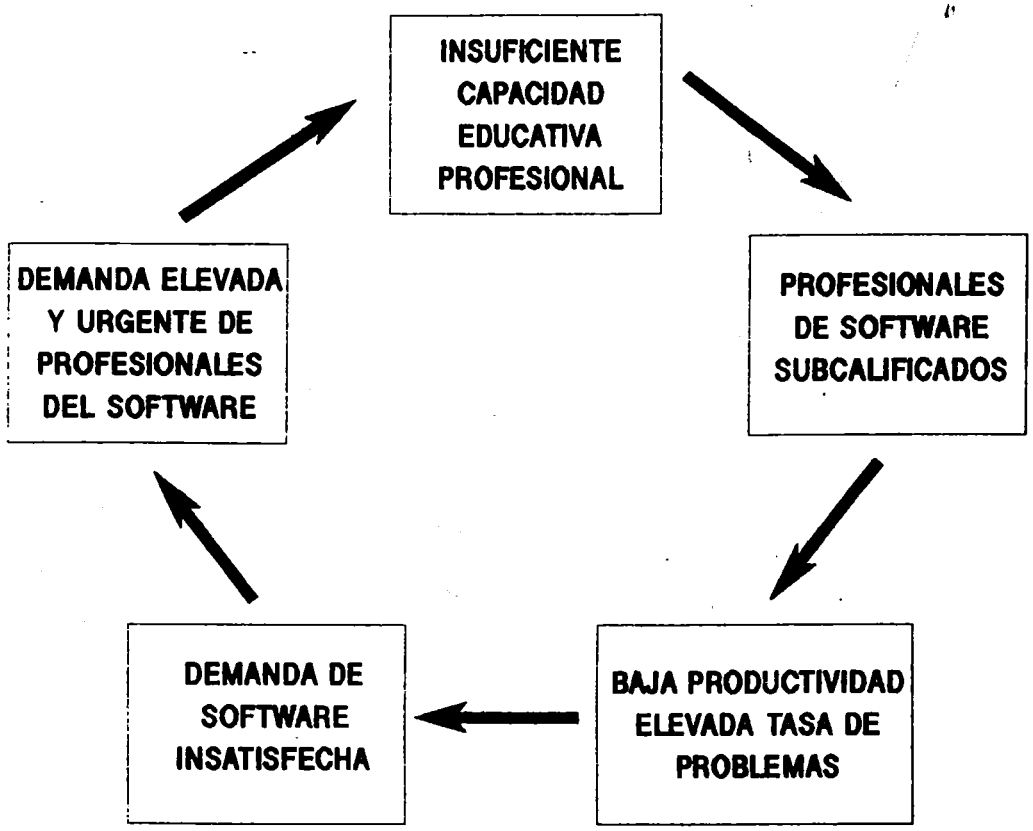
(WOLVERTON, 1974)

EL FACTOR HUMANO

COMO EJEMPLO DE LA INFLUENCIA DEL FACTOR HUMANO, EXTRAEMOS UN CUADRO CON CIERTOS RANGOS DE INFLUENCIA DE ALGUNOS ATRIBUTOS SOBRE LA PRODUCTIVIDAD EN EL DESARROLLO DE SOFTWARE **(SEGUN BOEHM, 1982).**

ATRIBUTO	RANGO, ENTE 1 Y:
-CAPACIDAD DEL EQUIPO HUMANO	4,18
-COMPLEJIDAD DEL PRODUCTO	2,36
-FIABILIDAD REQUERIDA	1,87
-EXPERIENCIA EN APLICACIONES SIMILARES	1,57
-LIMITACIONES DE MEMORIA	1,56
-TECNICAS MODERNAS DE PROGRAMACION	1,51
-HERRAMIENTAS DE SOFTWARE	1,49
-EXPERIENCIA DE LENGUAJE	1,20

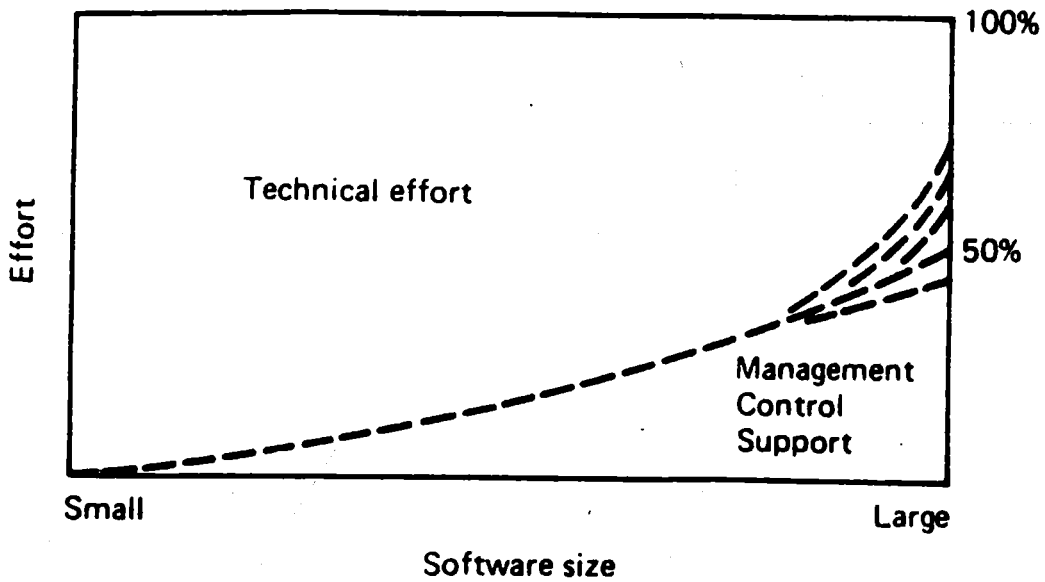
SIGNIFICACION: UN EQUIPO DE PROGRAMADORES Y ANALISTAS SITUADO EN EL PERCENTIL 90 DE CAPACIDAD MOSTRARA UNA PRODUCTIVIDAD, MEDIDA EN LINEAS DE CODIGO FUENTE POR HOMBRE-MES, UNAS 4 VECES SUPERIOR A LA DE OTRO EQUIPO SITUADO EN EL PERCENTIL 15.



CIRCULO VICIOSO PARA EL DESMORONAMIENTO DE LA CALIDAD DEL SOFTWARE
(Baber, 1982)

SOFTWARE BACKLOG DE 4 AÑOS EN MUCHOS SECTORES

(BOEHM)



Technical vs. support effort percent by scale.

REPRODUCCION PROHIBIDA

GESTION CICLO VIDA

- EL SOFT, ES COMO UN ROMPECABEZAS EN EL QUE HAY QUE DEFINIR EL DIBUJO, LOS COLORES, Y RECORTAR LAS PIEZAS, JUNTARLAS Y VOLVER A EMPEZAR CON OTRO DIBUJO.

TEORIA DEL PUZZLE BORROSO

- PRODUCIDO POR GRUPOS COORDINADOS DE PERSONAS.

LEYES DE LA NATURALEZA HUMANA

- TECNICAS VALIDAS PARA PROGRAMAS DEJAN DE SERLO PARA SISTEMAS DE PROGRAMAS (DIJKSTRA Y OTROS).

PRINCIPIO ELEMENTAL DE TEORIA SISTEMAS

- A MAYOR COMPLEJIDAD DEL SOFT, MAYOR COSTE Y MAYOR DESASTRE EN CASO DE FALLO.

ARGUMENTO A FAVOR DE LA MAXIMA DISTRIBUCION DE FUNCIONES

EL RETO DE LA DIRECCION DE PROYECTOS DE SOFTWARE(S.E.P.M)

(1)

Encuesta realizada en 1980 por Thayer, Pyster y Wood. Alos encuestados se les preguntó cuáles de entre una lista de 20 cuestiones constituían un problema y, en caso positivo, cómo de crítico era. También se les pidió que definieran si el problema era de naturaleza gerencial o técnica y que sugirieran una solución potencial. (Se aceptó que una determinada cuestión constituía un problema cuando así era votado por más de un 70% de encuestados; los encuestados se dividieron en 5 categorías distintas).

Twenty hypothesized problems in SEPM

Planning

1. *Requirements:* Requirement specifications are frequently incomplete, ambiguous, inconsistent, and/or unmeasurable.

2. *Success:* Success criteria for a software development are frequently inappropriate, which result in "poor-quality" delivered software; i.e., not maintainable, unreliable, difficult to use, relatively undocumented, etc.

3. *Project:* Planning for software engineering projects is generally poor.

4. *Cost:* The ability to estimate accurately the resources required to accomplish a software development is poor.

5. *Schedule:* The ability to estimate accurately the delivery time on a software development is poor.

6. *Design:* Decision rules for use in selecting the correct software design techniques, equipment, and aids to be used in designing software in a software engineering project are not available.

7. *Test:* Decision rules for use in selecting the correct procedures, strategies, and tools to be used in testing software developed in a software engineering project are not available.

8. *Maintainability:* Procedures, techniques, and strategies for designing maintainable software are not available.

9. *Warranty:* Methods to guarantee or warranty that the delivered software will "work" for the user are not available.

10. *Control:* Procedures, methods, and techniques for designing a project control system that will enable project managers to successfully control their project are not readily available.

Organizing

11. *Type:* Decision rules for selecting the proper organizational structure, e.g., project, matrix, function, are not available.

12. *Accountability:* The accountability structure in many software engineering projects is poor, leaving some question as to who is responsible for various project functions.

Staffing

13. *Project manager:* Procedures and techniques for the selection of project managers are poor.

Directing

14. *Techniques:* Decision rules for use in selecting the correct management techniques for software engineering project management are not available.

Controlling

15. *Visibility:* Procedures, techniques, strategies, and aids that will provide visibility of progress (not just resources used) to the project manager are not available.

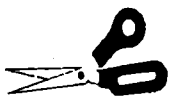
16. *Reliability:* Measurements or indexes of reliability that can be used as an element of software design are not available and there is no way to predict software failure; i.e., there is no practical way to show the delivered software meets a given reliability criteria.

17. *Maintainability:* Measurements or indexes of maintainability that can be used as an element of software design are not available; i.e., there is no practical way to show that a given program is more maintainable than another.

18. *Goodness:* Measurements or indexes of "goodness" of code that can be used as an element of software design are not available; i.e., there is no practical way to show that one program is better than another.

19. *Programmers:* Standards and techniques for measuring the quality of performance and the quantity of production expected from programmers and data processing analysts are not available.

20. *Tracing:* Techniques and aids that provide an acceptable means of tracing a software development from requirements to completed code are not generally available.



INTRODUCCION VISTA

EL RETO DEL S.E.P.M.

(SOFTWARE ENGINEERING PROJECT MANAGEMENT)

(2)

PROBLEMAS IMPORTANTES PERCIBIDOS

PLANIFICACION ↓ — UNANIMIDAD ↓

☆ REQUERIMIENTOS DEL SISTEMA	97%
☆ CRITERIOS DE EXITO EN EL DESARROLLO DE SW	82
☆ PROYECTO DE DESARROLLO	90
☆ ESTIMACION DE COSTES	88
☆ ESTABLECIMIENTO CALENDARIO	94
☆ METODOS GARANTIA	74
☆ SELECCION TECNICAS DE DISEÑO	72
☆ SELECCION TECNICAS DE TEST	79
☆ ESTRUCTURA DE RENDICION DE CUENTAS Y RESPONSABILIDADES (UNANIMIDAD)	81
☆ SELECCION JEFE PROYECTO (UNANIMIDAD)	77
☆ TECNICAS DE CONTROL DE FIABILIDAD DEL SW (UNANIMIDAD)	85
☆ TECNICAS DE CONTROL DE MANTENIBILIDAD	76
☆ TECNICAS Y ESTANDARES DE MEDIDA DE LA CANTIDAD/CALIDAD DE PRODUCCION DE PROGRAMADORES Y ANALISTAS (UNANIMIDAD)	78

EL RETO DEL S.E.P.M.

(3)

INTRODUCCION INTUITIVA

Table 1.
Categories for survey participants.

CATEGORY	DEFINITION
PROJECT MANAGERS	Managers of a project, including software line managers, each of whom had a firsthand knowledge of the major SEPM problems.
PROJECT INDIVIDUALS	Individual programmers/analysts who worked on projects.
TECHNICAL LEADERS	Individuals in high positions of influence in their companies' data processing function; highly visible authors or speakers on computer science, particularly data processing management; and authors of texts, papers, or reports on project management.
R&D PERSONNEL	Individuals who, through their business or avocation, were interested in furthering the state of the art in project management or, in a few cases, some other aspect of computer science.
EDUCATORS	Usually university professors; however, some educators from other educational institutes were included.

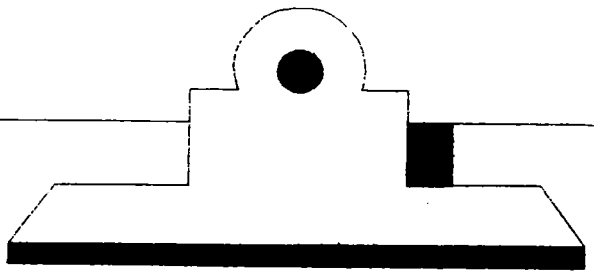
CINCO CATEGORIAS DE ENCUESTADOS

Table 2.
Summary of Part A results: Importance of problem.

MAJOR ISSUES	PERCENTAGE "PROBLEM" BY PARTICIPANT GROUP: COMPOSITE	PROJ MGRS	PROJ INDS	TECH LDRS	R&D	EDUCATORS
1. PLAN REQUIREMENT	97%	100%	97%	96%	94%	93%
2. PLAN SUCCESS	82	75	86	83	88	80
3. PLAN PROJECT	90	92	85	89	91	92
4. PLAN COST	88	92	86	84	88	95
5. PLAN SCHEDULE	94	97	93	91	92	95
6. PLAN DESIGN	72	71	67*	66*	72	75
7. PLAN TEST	79	85	77	73	69*	86
8. PLAN MAINTAINABILITY	67*	68*	64*	60*	70	72
9. PLAN WARRANTY	74	72	70	72	70	78
10. PLAN CONTROL	61*	54*	60*	54*	63*	69*
11. ORGANIZATIONAL TYPE	46*	51*	40*	42*	44*	48*
12. ORGANIZATIONAL ACCOUNTABILITY	81	75	72	86	89	88
13. STAFF PROJECT MANAGER	77	73	82	72	70	77
14. DIRECTING TECHNIQUES	59*	58*	50*	48*	57*	66*
15. CONTROL VISIBILITY	58*	65*	62*	71	77	74
16. CONTROL RELIABILITY	85	90	84	78	79	86
17. CONTROL MAINTAINABILITY	76	76	67*	71	82	82
18. CONTROL GOODNESS	62*	60*	62*	60*	56*	65*
19. CONTROL PROGRAMMERS	78	77	70	78	81	84
20. CONTROL TRACING	67*	67*	55*	70*	69*	68*

*Results under 70% were inconclusive

DETALLE DE LAS RESPUESTAS DE LOS CINCO GRUPOS



CATEGORIAS DE SOFTWARE

- ☆ TAMAÑO DEL SOFTWARE
- ☆ SOFTWARE DEL SISTEMA, DE SOPORTE, DE APLICACION
- ☆ SOFTWARE-PROYECTO Y SOFTWARE-PRODUCTO
- ☆ TIPOLOGIA DE BROOKS
- ☆ SOFTWARE HORIZONTAL, VERTICAL Y GENERICO
- ☆ SOFTWARE INTEGRADO
- ☆ CLASIFICACION S, P, E DE LOS PROGRAMAS

SOFTWARE GRANDE: $S > 75.000$ SENTENCIAS FUENTE

SOFTWARE MEDIANO: $18.000 < S < 75.000$ "

SOFTWARE PEQUEÑO: $S < 18.000$ "

TAMAÑO DEL SOFTWARE

(PUTNAM, 1982)

SOFTWARE SISTEMA O DE BASE

SISTEMA OPERATIVO

SISTEMA GESTION B.DATOS

SISTEMA COMUNICACIONES

SOFTWARE SOPORTE O DESARROLLO

COMPILADORES

ENSAMBLADORES

GENERADORES

EDITORES

CONVERSORES DE SOPORTES

ANALIZADORES

SIMULADORES

GENERADORES DE PRUEBAS

CONSTRUCTORES DE GRAFICOS

PERT

LIBRERIAS

- - - -

SOFTWARE APLICACION

- - - -

TRES CATEGORIAS DE SOFTWARE

* SOFTWARE-PRODUCTO ⟶ MUCHOS Y DIFERENTES USUARIOS E INSTALACIONES

* SOFTWARE-PROYECTO ⟶ UNO O MUY POCOS USUARIOS E INSTALACIONES

⇓

EL ESFUERZO EN EL CICLO DE VIDA ES MUY DIFERENTE

PARA SOFTWARE-PRODUCTO:

- DOCUMENTACION MUY CUIDADA, PARA VENTAS, USUARIOS, MANTENIMIENTO
- MODULARIZAR AL MAXIMO
- VERIFICAR EXHAUSTIVAMENTE EL CODIGO
- PROBAR CODIGO CON TODOS LOS SOFTW. SISTEMA PREVISTOS
- PREPARAR INTERFAZ HOMBRE-MAQUINA
- ESTIMAR Y PLANIFICAR PRESUPUESTOS, ESFUERZOS DE CAMBIO, ADAPTACIONES Y VERSIONES FUTURAS
- ESTABLECER SISTEMA RECOGIDA INCIDENTES Y ERRORES Y SISTEMA DE DISTRIBUCION DE MODIFICACIONES Y VERSIONES

2 CLASES DE SOFTWARE, SEGÚN NÚMERO Y VARIEDAD DE USUARIOS

TIPOLOGIA DE BROOKS

(BROOKS, 1982)

PROGRAMA

PROGRAMA - PRODUCTO

COMPONENTE DE SOFTWARE - PRODUCTO

SOFTWARE - PRODUCTO

EL "PROGRAMA", LISTO PARA SER EJECUTADO POR SU PROPIO AUTOR Y SOBRE EL SISTEMA PARA EL QUE FUE DESARROLLADO. ESTA ES LA MODALIDAD QUE CORRESPONDE AL OBJETO QUE PRODUCE EL PROGRAMADOR INDIVIDUAL PARA SU USO. SU ESFUERZO HAY QUE MULTIPLICARLO POR TRES PARA CONVERTIRLO EN UNA DE LAS DOS MODALIDADES SIGUIENTES Y POR NUEVE PARA LLEVARLO A LA ÚLTIMA MODALIDAD.

EL "PROGRAMA-PRODUCTO". ES UN PRODUCTO QUE PUEDE SER EJECUTADO, VERIFICADO Y AMPLIADO POR CUALQUIERA EN DISTINTOS ENTORNOS OPERATIVOS Y PARA MUCHOS CONJUNTOS DE DATOS.

EL "COMPONENTE DE SOFTWARE-PRODUCTO", SIENDO ÉSTE UNA COLECCIÓN DE PROGRAMAS QUE FORMAN UN CONJUNTO, COORDINADOS EN FUNCIÓN Y DISCIPLINADOS EN SU FORMATO, DE MANERA QUE SU ENSAMBLAJE CONSTITUYA UNA TOTALIDAD OPERATIVA. ESTE PROGRAMA HA DE SER DISEÑADO CON SUS ENTRADAS Y SALIDAS CONFORMES SINTÁCTICA Y SEMÁNTICAMENTE RESPECTO DE INTERFACES MUY PRECISAMENTE DEFINIDAS Y CON SUS CARACTERÍSTICAS SOMETIDAS A DETERMINADAS LIMITACIONES DE MEMORIA, PERIFERIA Y TIEMPO DE PROCESAMIENTO. EL PROGRAMA NECESITA SER PROBADO EN RELACIÓN CON OTROS COMPONENTES DEL SISTEMA BAJO TODAS LAS COMBINACIONES PREVISTAS.

EL "SOFTWARE-PRODUCTO", QUE RESULTA DE LA COMBINACIÓN DE LOS DOS ÚLTIMOS.

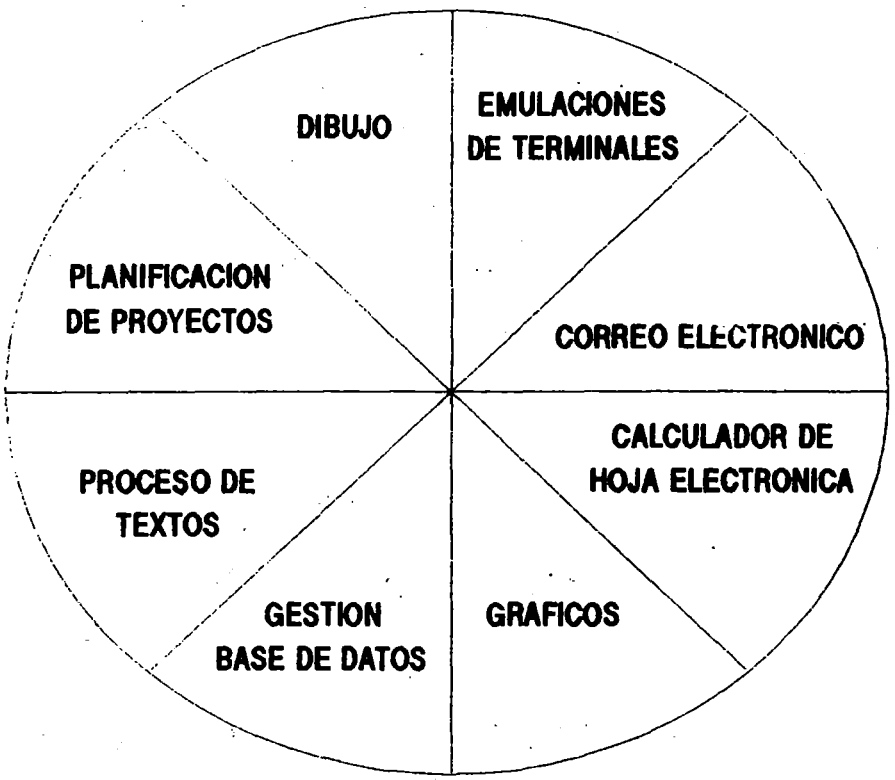
PROGRAMAS HORIZONTALES, VERTICALES Y GENERICOS

EL CALIFICATIVO DE HORIZONTAL (O FUNCIONAL) SE DEBE AL "ANCHO ESPECTRO DE POSIBLES USUARIOS DE UNA POBLACIÓN", VERBIGRACIA: CONTABILIDAD, NÓMINAS, ALMACÉN. SUELEN SER MODULARES, PARA PERMITIR UNA CIERTA PERSONALIZACIÓN A CADA CASO, CON MÍNIMA INTERVENCIÓN DE SU AUTOR.

PROGRAMAS DE APLICACIÓN VERTICAL (O SECTORIAL) SON PAQUETES ORIENTADOS A UNA CLASE DE USUARIOS, COMO PUEDEN SER INGENIEROS DE ESTRUCTURAS, NOTARIOS U ODONTÓLOGOS.

LAS HOJAS ELECTRÓNICAS, LOS PROCESADORES DE TEXTO, LOS SISTEMAS DE BASES DE DATOS SE CONSIDERAN PROGRAMAS GENÉRICOS. A SEMEJANZA DE LOS PROGRAMAS HORIZONTALES TIENEN UN AMPLIO USO, PERO MÁS QUE PARA UNA APLICACIÓN EN SÍ, POR GENERAL QUE ÉSTA SEA, SE NOS OFRECEN COMO HERRAMIENTAS AL SERVICIO DE QUEHACERES MUY COMUNES.

SOFTWARE INTEGRADO



MAXIMO DE FUNCIONES PREVISTAS EN UN PAQUETE DE SOFTWARE INTEGRADO, EN 1985 (PERTIG, 1985, p.24)

EN EL FUTURO SE AÑADIRAN MANIPULADORES INTELIGENTES DE SIMBOLOS: SISTEMAS EXPERTOS, SINTETIZADORES Y RECONOCEDORES DE VOZ, PROGRAMAS DE COMPRENSION Y TRADUCCION DE LENGUAJE NATURAL, SIMULADORES, ETC.

CATEGORIZACION SPE DE LOS PROGRAMAS
(SEGÚN ÍNDICES PREVISIBLES DE CAMBIO)
(TURSKI, 1979) (LEHMAN, 1980)

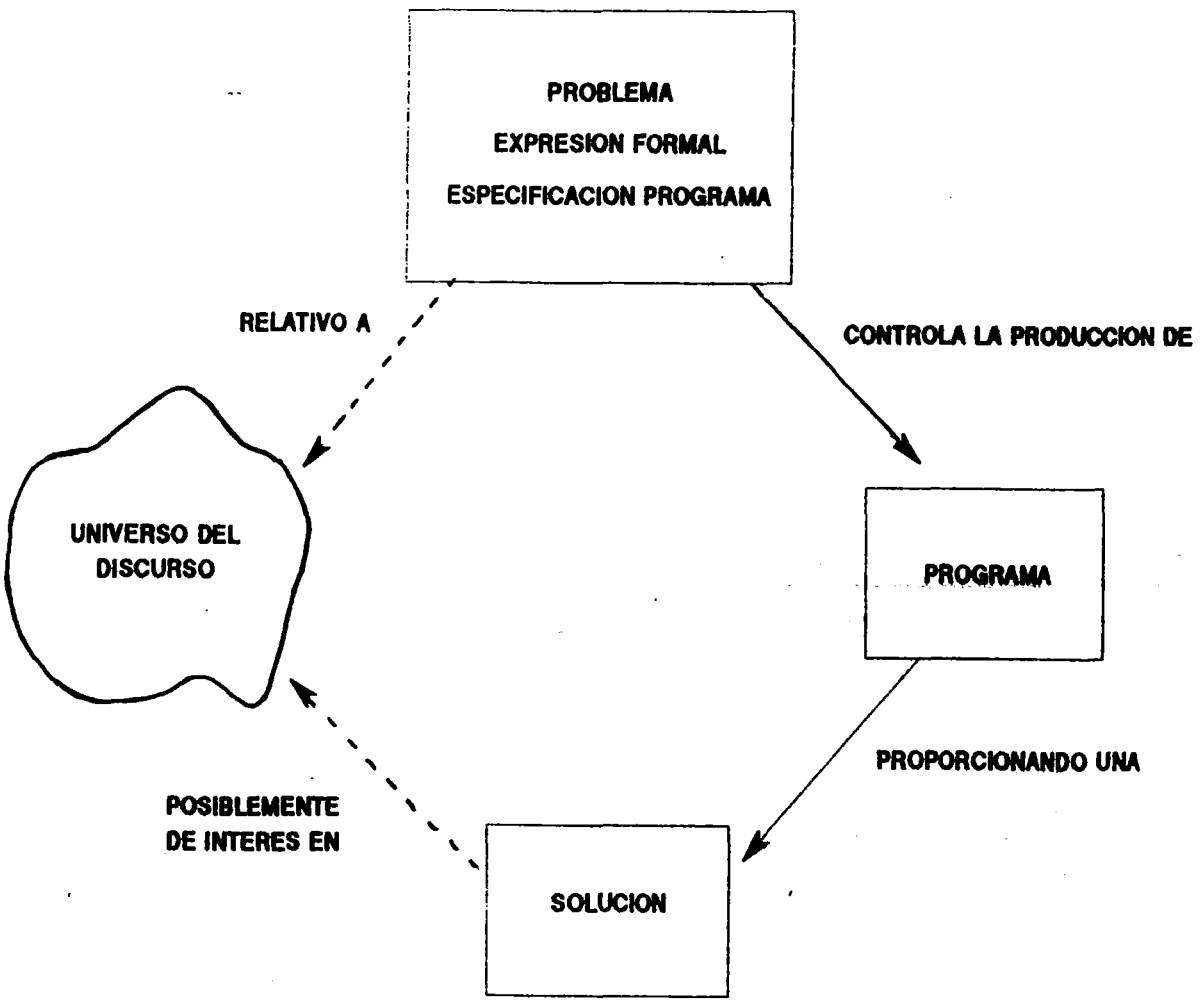
S.PROGRAMAS Specification program.
(Ej. los filósofos comiendo)

Completamente definido
Programa conceptualmente estático
Correctitud basada en especificaciones

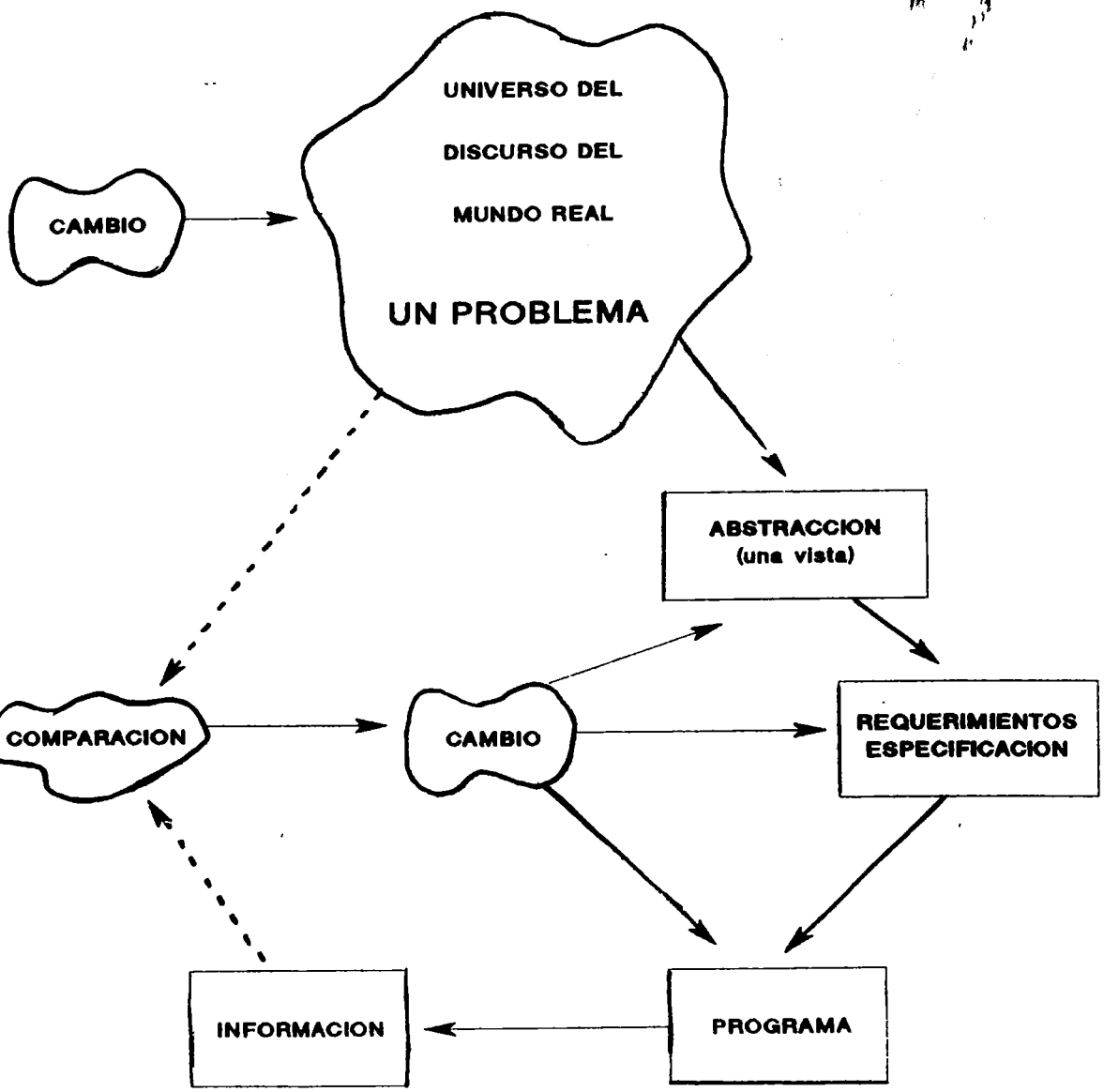
P.PROGRAMAS Real World Problem Program
(Ej. Predicción del tiempo atmosférico)

Soluciones aproximadas
Abstracción con incertidumbres, criterios, incognitas, decisiones del analista
Bucle de realimentación basado en la aceptabilidad de la solución en contexto mundo real.
Correctitud ⟶ Validez
El cambio no genera un nuevo problema, sino una percepción distinta

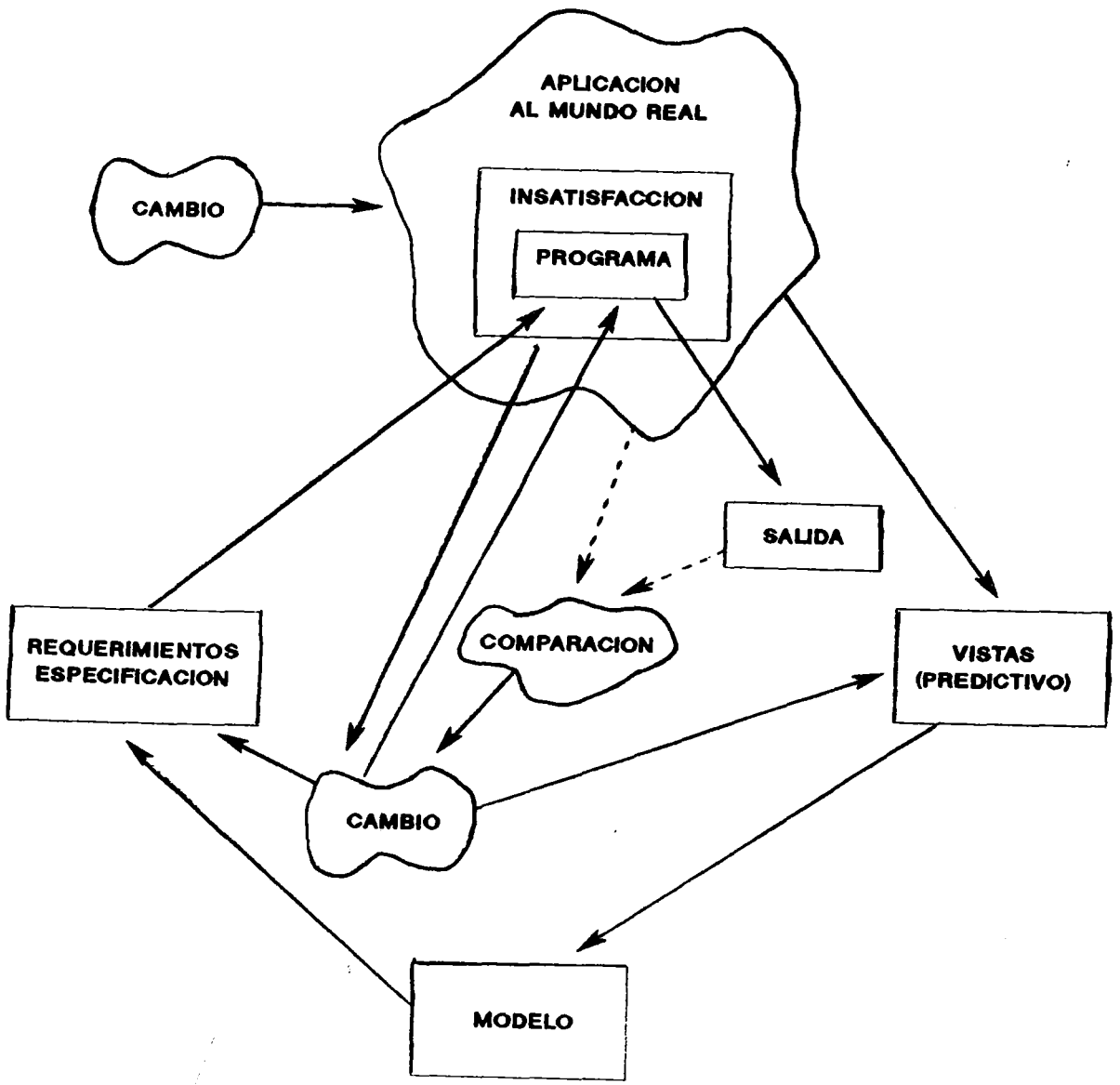
E.PROGRAMS ¿Environment? program
(Ej. Sistema Operativo)
Mecanizan actividades humanas o sociales
Forman parte del mundo real que modelan
Extrapolación y predicción de consecuencias (subjetividad)
Comportamiento cambiante de los usuarios con el uso
Entorno cambiante del Sistema



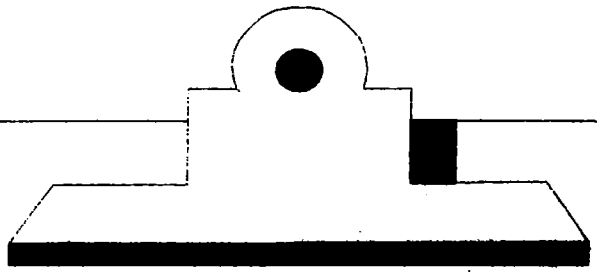
S-PROGRAMAS
(CLASIFICACION SPE)



P-PROGRAMAS
(CLASIFICACION SPE)



E-PROGRAMAS
(CLASIFICACION SPE)



MODELOS 3P Y 5P DE LOS PROYECTOS DE SOFTWARE. APROXIMACION CIBERNETICA Y CICLO DE VIDA TRIFASICO

- ☆ PROBLEMA, PRODUCTO Y PROCESO
- ☆ EXPERIMENTO DE WEINBERG – SCHULMAN
- ☆ TASA DE CAMBIO Y BORROSIDAD
- ☆ REPRESENTACION CIBERNETICA DEL MODELO 3P
- ☆ CICLO DE VIDA TRIFASICO
- ☆ TECNICAS PREDOMINANTES SEGUN LAS FASES
- ☆ MODELO 5P

MODELO EP

PROBLEMA ⟶ PROCESO ⟶ PRODUCTO

⟲

→ ANALISIS DE LA NATURALEZA DEL **PROBLEMA**

(MODELACIÓN INFORMACIONAL)

↓

ANALISIS DE LA NATURALEZA DEL SOFTWARE

→ (DETERMINACIÓN DE LAS CARACTERÍSTICAS

BASICAS DEL **PRODUCTO**)

↓

DISEÑO GLOBAL DEL **PROCESO**

↓

CONSTRUCCION DEL SOFTWARE

(**PROCESO** DE DESARROLLO)

↓

ANALISIS DEL SOFTWARE

(**PRODUCTO** DEL PROCESO)

RANGO DE CLASIFICACION DE RESULTADOS (1: MEJOR RESULTADO)

OBJETIVO A OPTIMIZAR POR EL EQUIPO:	ESFUERZO	NUMERO DE SENTENCIAS	OCUPACION DE MEMORIA	CLARIDAD DEL PROGRAMA	CLARIDAD DE SALIDAS
ESFUERZO	1	4	4	5	3
NUMERO DE SENTENCIAS	2-3	1	2	3	5
OCUPACION DE MEMORIA	5	2	1	4	4
CLARIDAD DEL PROGRAMA	4	3	3	2	2
CLARIDAD DE SALIDAS	2-3	5	5	1	1

EXPERIMENTO DE WEINBERG - SCHULMAN (PROCESO ⟷ PRODUCTO)

MODELO SP

TASA DE CAMBIO Y BORROSIDAD

TASA DE CAMBIO: LA DEMANDA DE CAMBIO A LO LARGO DEL TIEMPO DE LA SOLUCION (PRODUCTO) DEL PROBLEMA ⟶ VER CLASIFICACION S.P.E.

BORROSIDAD DEL PROBLEMA: DIFICULTAD PARA DEFINIR UN SISTEMA, UN MODELO O LAS AREAS CONCRETAS DE SOLUCION DEL PROBLEMA.

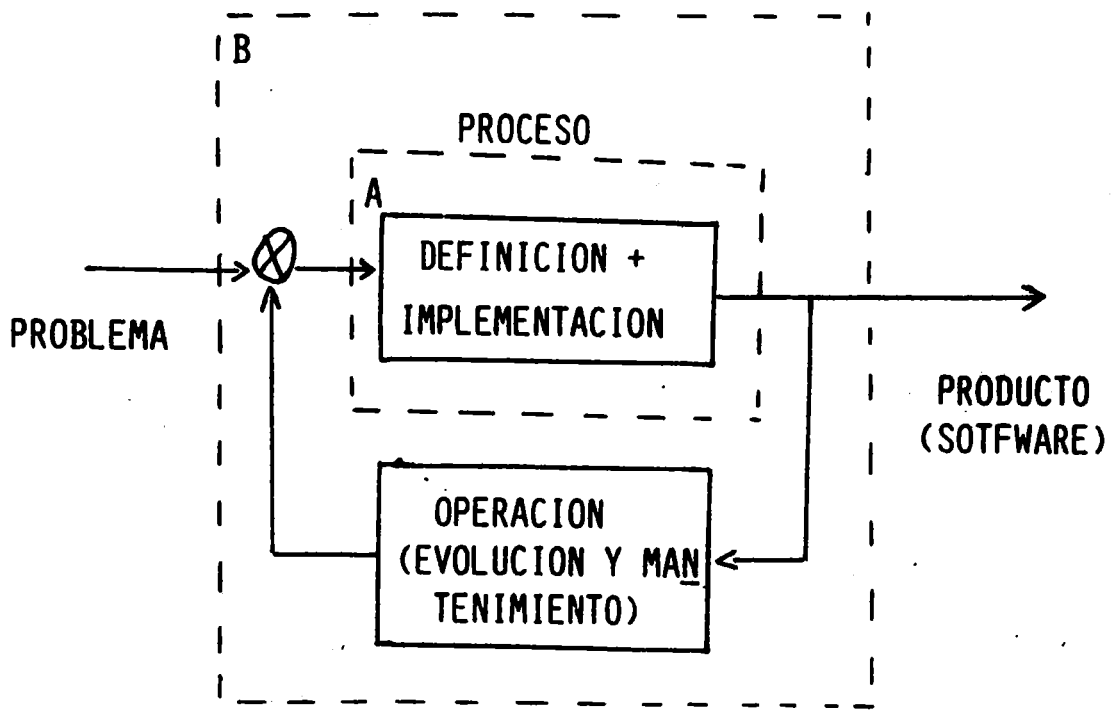
ES UN ATRIBUTO DEL PROBLEMA Y UN ATRIBUTO - DEL DISEÑADOR FRENTE A ESE PROBLEMA:

PROBLEMA ⟷ DISEÑADOR

⇩

BORROSIDAD

APROXIMACION CIBERNETICA



ENFOQUE A

ENFOQUE B. BIFASICO: NECESARIO. SI HAY CICLO DE VIDA

MODELO 5P

CICLO TRIFASICO DE VIDA

- FASES DOMINANTES -

FASES DEL PROCESO (DISEÑO GLOBAL)	CARACTERISTICAS DEL PROBLEMA (TASA DE CAMBIO, BORROSIDAD)			
	00	10	01	11
1A. FASE	--	--	X	X
2A. FASE	X	X	X	X
3A. FASE	--	X	--	X

1A. FASE: DEFINICION DEL SISTEMA (ESPECIFICACION, DISEÑO)

2A. FASE: IMPLEMENTACION (DISEÑO PROGRAMAS, CODIFICACION, PRUEBA, INTEGRACION)

3A. FASE: MANTENIMIENTO Y EVOLUCION (CORRECCION DE ERRORES, CAMBIOS, NUEVAS VERSIONES Y MEJORAS)

MODELO SP

TECNICAS PREDOMINANTES SEGUN LAS FASES

1ª. FASE: ANALISIS Y DISEÑO DE SISTEMAS
RESOLUCION DE PROBLEMAS
TOMA DE DECISIONES
ES UN CAMPO EN EL QUE APENAS EXISTEN TECNICAS Y LENGUAJES FORMALIZADOS CON CARACTER GENERAL. SI EXISTEN, TIENEN UN DOMINIO APLICATIVO ESTRECHO.

2ª. FASE: TECNICAS DE SOFTWARE EN SENTIDO ESTRICTO.
DISEÑO Y PROGRAMACION ESTRUCTURADA.
MODULARIZACION.
HERRAMIENTAS DE AYUDA.
ENTORNOS DE PROGRAMACION.
RAPIDA EVOLUCION.

3ª. FASE: RESULTADO DE 1ª. Y 2ª. FASE. UTILIZACION DE AMBOS TIPOS DE TECNICAS.
NECESIDAD DE HABER HECHO ENFASIS EN PLANIFICAR EL PROCESO PARA CONSEGUIR UN PRODUCTO CON SIGUIENTES CRITERIOS DE CALIDAD: DOCUMENTACION, MODULARIDAD, FIABILIDAD, MODIFICABILIDAD, CLARIDAD, MANTENIBILIDAD.

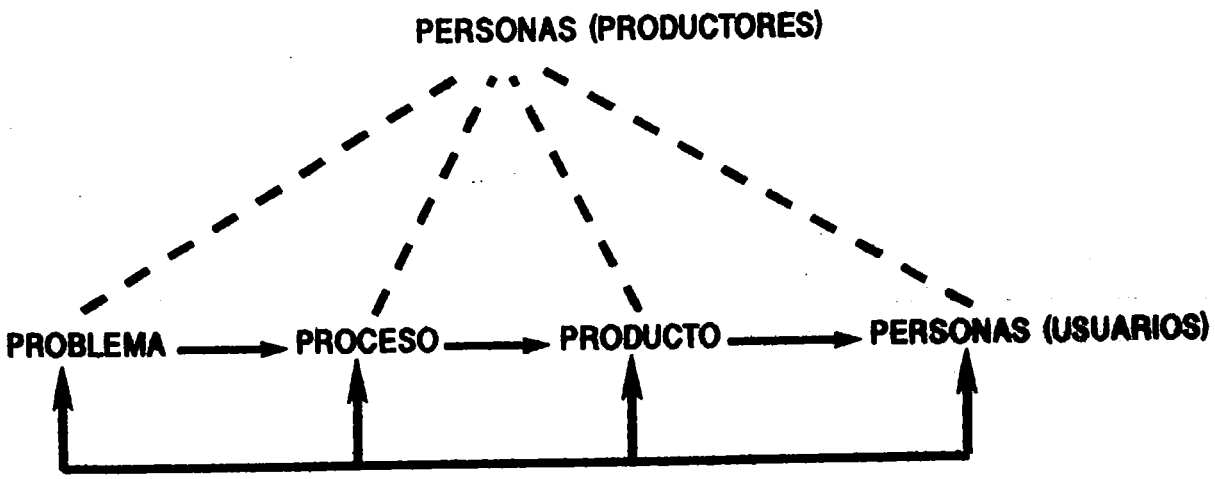
MODELO 5P

TECNICAS DE GESTION DE PROYECTOS

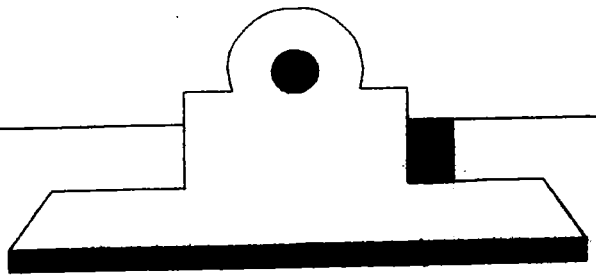
ENVUELVEN Y COORDINAN EL **CONJUNTO** DE LAS TECNICAS DE **TODAS LAS FASES.**

INVOLUCRAN:

- ASPECTOS TECNICOS
- ASPECTOS ECONOMICOS
- ASPECTOS ORGANIZATIVOS
- ASPECTOS HUMANOS

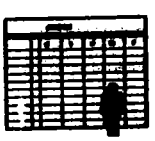


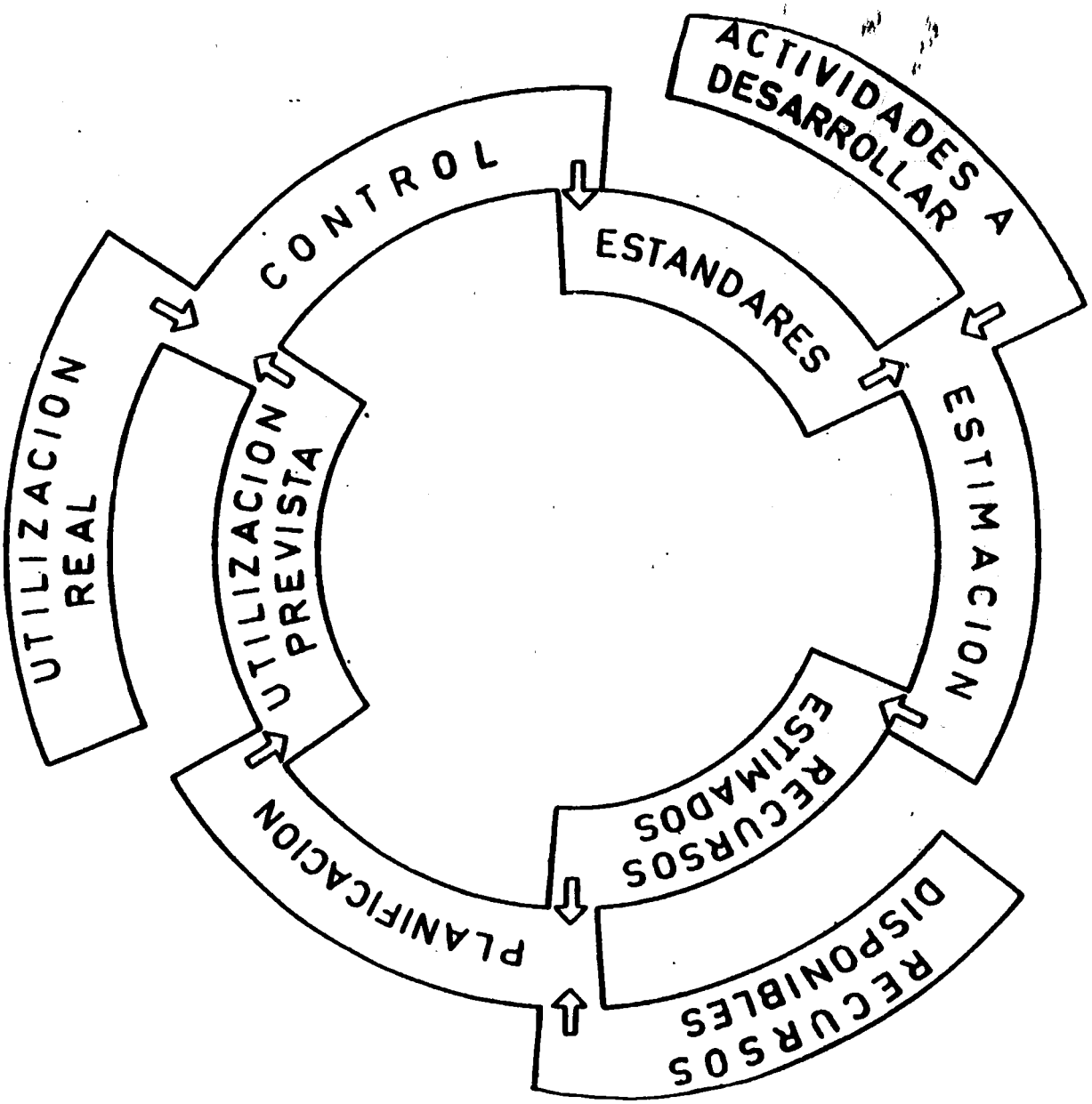
MODELO 5P EN LA INGENIERIA DEL SOFTWARE



TECNICAS DE ESTIMACION

- ☆ EL CICLO ESTIMACION-PLANIFICACION-CONTROL
- ☆ MICROESTIMACION Y MACROESTIMACION
- ☆ TECNICA CLASICA DE PLANIFICACION DE RECURSOS HUMANOS/CONTINGENCIAS
- ☆ CONCEPTO DE PRODUCTIVIDAD EN SOFTWARE
- ☆ CONFLICTO DE CANTIDAD Y CALIDAD
  - -MODULOS PROPENSOS A DEFECTOS
- ☆ EVOLUCION TECNOLOGICA DE LA PRODUCTIVIDAD





EL CICLO ESTIMACION→PLANIFICACION→CONTROL

-TECNICAS DE ESTIMACION-

MICROESTIMACION. SE FIJAN TAMAÑO, FECHA INICIAL Y DURACION DE CADA ACTIVIDAD DISTINGUIBLE, SE PRACTICAN AJUSTES PARA CALIBRAR PERSONAL, COMPLEJIDAD, INCERTIDUMBRES Y OTROS FACTORES (IDENTIFICADOS UNOS 100 FACTORES, ENTRE ELLOS 42 SIGNIFICATIVOS Y UNOS 29 CUANTIFICABLES).

MACROESTIMACIONES. DADOS CIERTOS DATOS ACERCA DE UN PROYECTO DE DESARROLLO DE SOFT, DISPONIBLES AL INICIO DEL DESARROLLO, EL MACROENFOQUE GENERA UNA CURVA TEMPORAL ESPERADA DEL ESFUERZO HUMANO A LO LARGO DEL CICLO.

ESTIMACION

TECNICA CLASICA DE PLANIFICACION DE RECURSOS HUMANOS

FASES EN EL DESARROLLO DE UN PLAN DE RECURSOS HUMANOS EN UN PROYECTO (ESTIMACION + PLANIFICACION).

1. ENUMERAR LAS TAREAS QUE COMPONEN EL TRABAJO DEL PROYECTO.

2. DETERMINAR DEPENDENCIA ENTRE TAREAS.

3. DETERMINAR RESTRICCIONES EXTERNAS.

4. ASIGNAR PERSONAS A TAREAS.

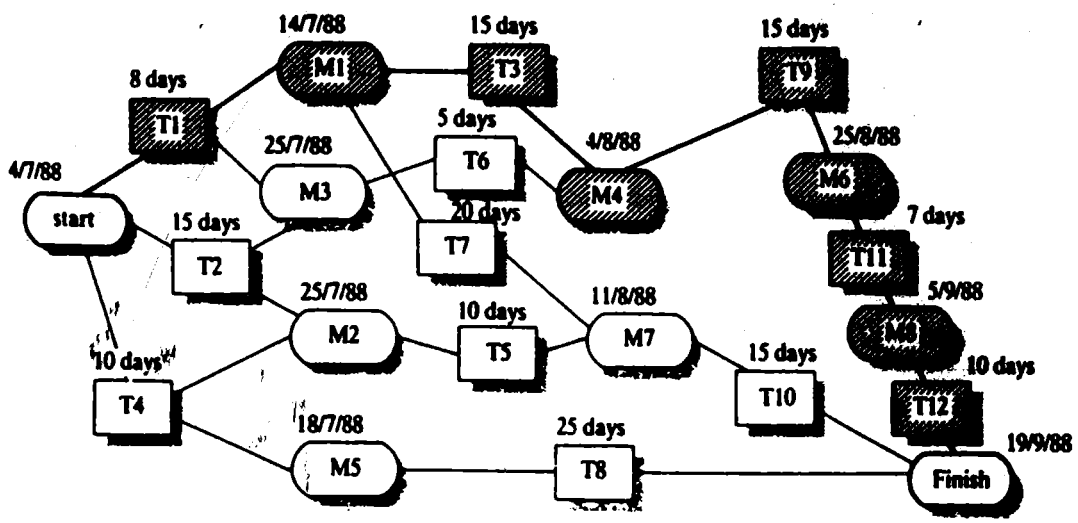
5. ESTIMAR TIEMPOS PARA CADA TAREA.

6. PROVEER PARA CONTINGENCIAS.

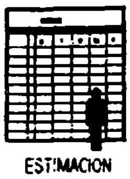
CRUCE DE FUNCION DEL SISTEMA A DESARROLLAR CON ACTIVIDAD (P. EJ. CODIFICAR, PROBAR) LA TAREA

Task	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1
T4	10	
T5	10	T2, T4
T6	5	T1, T2
T7	20	T1
T8	25	T4
T9	15	T3, T6
T10	15	T5, T7
T11	7	T9
T12	10	T11

TAREAS, DURACION Y DEPENDENCIAS
(SOMMERVILLE, 1988)



RED DE ACTIVIDADES
(SOMMERVILLE, 1988)



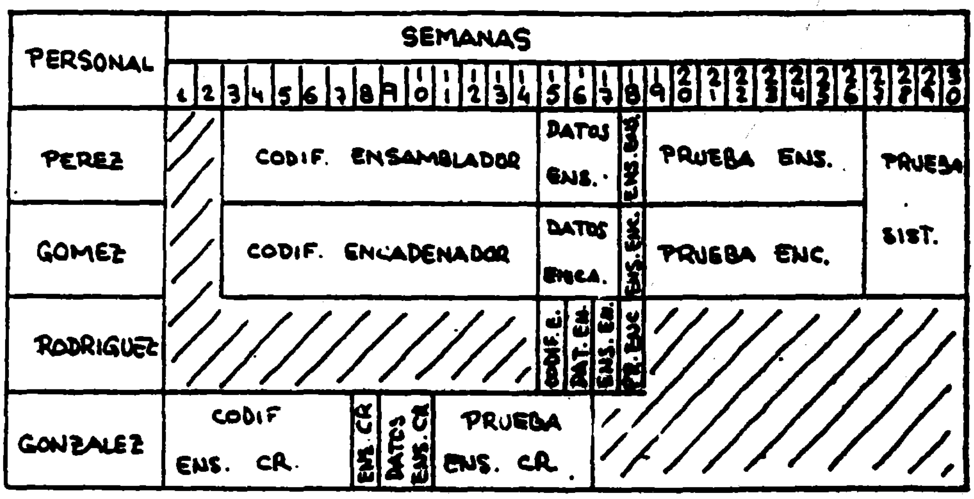


DIAGRAMA DE BARRAS

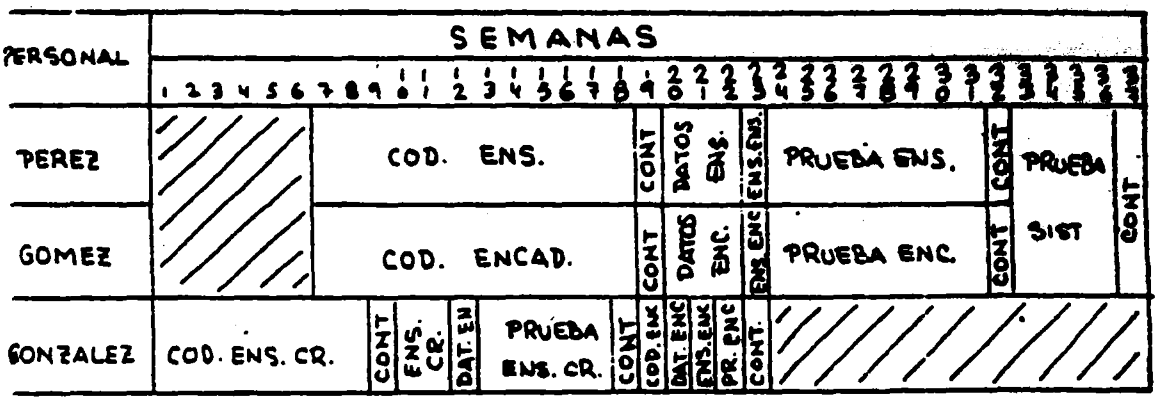


DIAGRAMA DE BARRAS CON CONTINGENCIAS POR TAREAS

CONTINGENCIAS POSIBLES

1. OBLIGACION TEMPORAL EXTRA. UN PARTICIPANTE EN EL PROYECTO ES TEMPORALMENTE REQUERIDO PARA UN ASUNTO ESPECIAL. POR EJEMPLO, SE LE REQUIERE PARA DETECTAR UN ERROR PREVIAMENTE NO LOCALIZADO EN UN PROGRAMA EN EXPLOTACIÓN.

2. APTITUDES INADECUADAS EN EL PERSONAL. LAS PERSONAS ASIGNADAS AL PROYECTO NO POSEEN EL GRADO DE APTITUDES NECESARIAS A LAS TAREAS ENCOMENDADAS.

3. TRANSFERENCIA. MOVIMIENTO A OTRO PROYECTO DE MAYOR PRIORIDAD DE UN PARTICIPANTE EN ÉSTE.

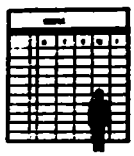
4. ENTREGA TARDIA. RECURSOS QUE NO LLEGAN EN EL MOMENTO ACORDADO. EJEMPLOS: PERSONAL, HARDWARE, SOFTWARE Y TIEMPO DE COMPUTADOR.

5. FALLOS EN SERVICIOS. LOS TIEMPOS DE RESPUESTA DE CIERTOS SERVICIOS SE DETERIORAN POR ENCIMA DE LOS NIVELES PLANIFICADOS. EJEMPLO : PRUEBAS EN COMPUTADOR.

6. APTITUDES INADECUADAS EN EL PERSONAL DE OPERACION. ES EL CASO DE FUNCIONES QUE NO SE LLEVAN A CABO CORRECTAMENTE POR MALA COMPRENSIÓN DE LOS OPERADORES (DE CONSOLA, DE PERIFERIA...).

7. FALLOS DE SOFTWARE. EL PROYECTO EMPLEA UN PROGRAMA NO DESARROLLADO EN EL PROYECTO Y SE DESCUBRE QUE TAL PROGRAMA NO FUNCIONA SEGÚN LAS ESPECIFICACIONES QUE DE ÉL SE POSEEN.

ESTA SITUACIÓN OBLIGA A GASTAR TIEMPO EN:



ESTIMACION

A). MODIFICAR DETERMINADOS PROGRAMAS DEL PROYECTO PARA SORTEAR LOS FALLOS DEL SOFT. Y DAR TIEMPO A - QUE ÉSTOS SEAN REPARADOS.

B). MODIFICAR ESOS PROGRAMAS DE FORMA QUE ÉSTOS PUEDAN OPERAR CON EL SOFT. UNA VEZ REVISADO EL MISMO.

8. FALLOS DE EQUIPO. EL COMPUTADOR O CUALQUIER OTRO - - EQUIPO, COMO PERIFÉRICOS O TERMINALES.

9. FALTA DE INFORMACION. POR EJEMPLO, INFORMACIONES TALES COMO DECISIONES Y POLÍTICAS IMPRESCINDIBLES PARA CONTINUAR EL PROYECTO A PARTIR DE UN INSTANTE O FASE PRECISA.

10. ROTACION. UNA PERSONA DEL PROYECTO ABANDONA ÉSTE POR PROMOCIÓN O POR CAUSAR BAJA EN LA EMPRESA.

11. AUSENCIA. ENFERMEDAD U OTRO TIEMPO LEGALMENTE AUTORIZADO O DEBIDAMENTE JUSTIFICADO.

12. REUNIONES DE LA EMPRESA AJENAS AL PROYECTO.

Sobre las fases de planificación:

1 y 2. CONOCIMIENTO DEL SISTEMA (?)

3. NO SIEMPRE PREVISIBLES Y MUCHAS VECES INCONTROLABLES.
Posibles: plazos, tiempos de respuesta ordenador, fechas entrega de un recurso.

4. PSICOLOGIA Y EXPERIENCIA
Rotación o cambios de categoría.
Defecto común: no asignar personas a tareas de control o de coordinación.

5. Factores mínimos a considerar: a) complejidad de la tarea, b) capacidad de la persona.
3 métodos principales: Juicio profesional
Técnica histórica
Estándares: siempre variables por los cambios tecnológicos y también metodológicos.

En general, un elevado desconocimiento del sistema a desarrollar puede impedir algo tan importante como una estimación global del esfuerzo (coste) requerido y del tiempo necesario.

Sólo se consideran las etapas del desarrollo, y más exactamente la etapa de programación. Esto muestra que aquí se mezcla estimación con planificación y que ha de haber estimación/planificación en las etapas de definición y de análisis y diseño, si los hubiera. Y ha de haber o puede haber estimación en la larga etapa del mantenimiento.

ESTIMACION

LAS TECNICAS DE ESTIMACION SE BASAN EN LA EXPERIENCIA

* **EXPERIENCIA PROPIA.**

* **EXPERIENCIA AJENA.**

* **ESTANDARES: EXPERIENCIA CONTROLADA Y CUANTIFICADA.**

FACTORES QUE MAS CONTRIBUYEN AL COSTE DEL DESARROLLO DE SOFTWARE (SEGUN EXPERIENCIA DE J. FOX, 1982)

* **ESCALA.** La cantidad de función a desarrollar.

* **CLARIDAD.** El grado de comprensión de las funciones a desarrollar.

* **COMPLEJIDAD LOGICA.** El número de ramas condicionales por cada 100 instrucciones.

* **CONSECUENCIAS DE FALLOS.** Cuánto diseño y esfuerzo hay que realizar para conseguir los requisitos de fiabilidad y recuperabilidad.

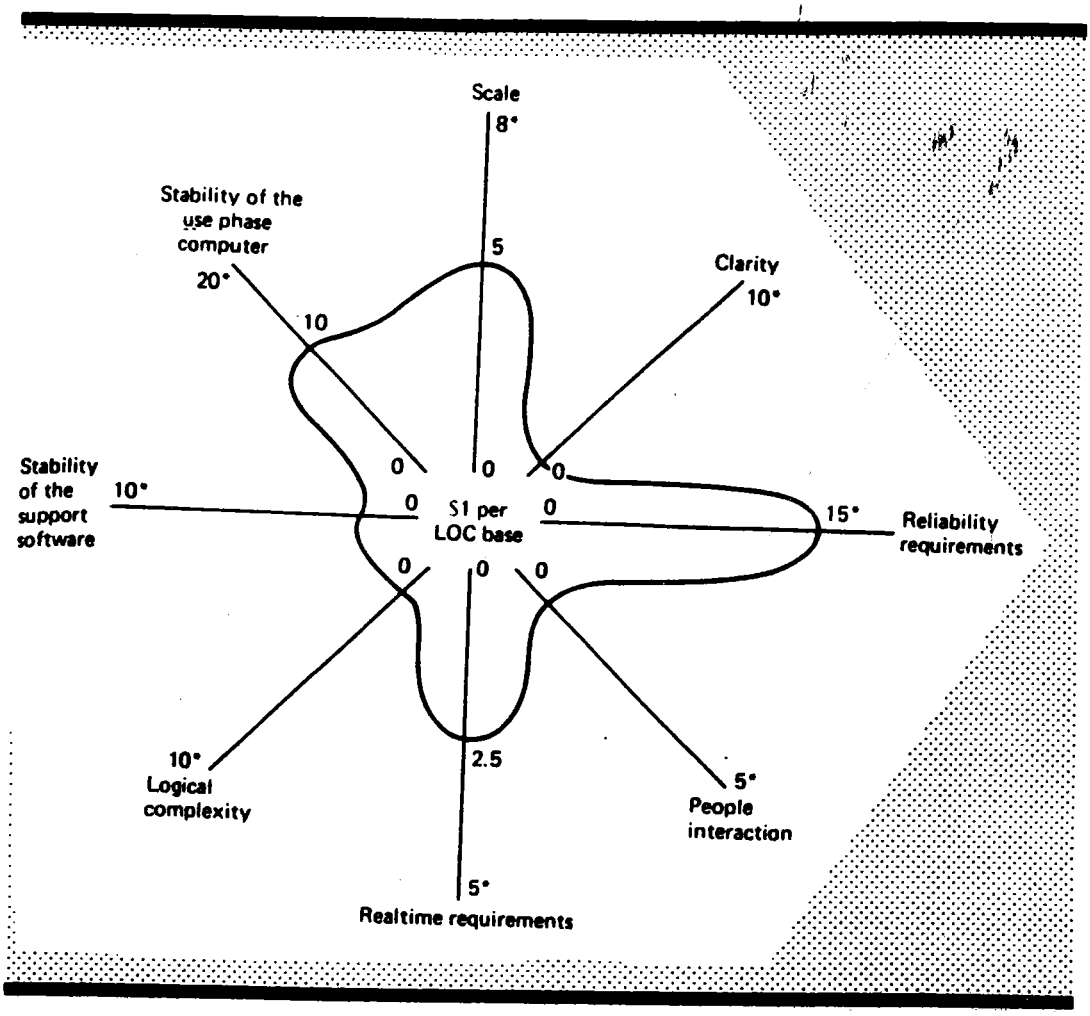
* **INTERACCION CON USUARIOS.** Cuán a menudo e intensiva es la interacción del usuario con el sistema.

* **REQUISITOS DE TIEMPO REAL.** Cuán rápidas deben ejecutarse las funciones.

* **ESTABILIDAD DEL SOFTWARE SOPORTE.** ¿Es el software soporte estable y maduro?.

* **ESTABILIDAD DE LOS ORDENADORES EN EXPLOTACION.**

ESTIMACION

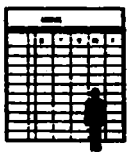


ESTIMACION BAJO EL SIGUIENTE

CUADRO DE CONDICIONES: $32.50

POR LINEA DE CODIGO (BASE $ 1)

*	SIN	MULTIPLICADOR	PARA COMPLEJIDAD LOGICA
*	"	"	CLARIDAD
*	"	"	INTERACCION USUARIOS
*	"	"	ESTABILIDAD SOFT. SOPORTE
*	x 5	"	ESCALA
*	x 2.5	"	TIEMPO REAL
*	x 15	"	EXIGENCIA DE FIABILIDAD
*	x 10	"	INESTABILIDAD ORDENADOR



CONCEPTO Y MEDIDA DE LA PRODUCTIVIDAD EN SOFTWARE

SIENDO LA PRODUCTIVIDAD LA CANTIDAD DE PRODUCTO POR UNIDAD DE RECURSO EMPLEADO, ES NECESARIO DEFINIRLA BIEN Y MEDIRLA PARA ESTIMAR, PLANIFICAR, CONTROLAR Y COMPARAR.

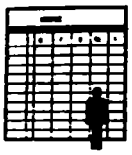
ALGUNOS ASPECTOS A CONSIDERAR EN LA DEFINICION

* AMBITOS DE APLICACION DEL CONCEPTO.
* UNIDADES DE MEDIDA: NIVEL DE ACABADO DEL SW.
* TIEMPOS.
* ELEMENTOS CONTABLES.

CONFLICTOS DE CANTIDAD Y CALIDAD (EXPERIMENTO DE WEINBERG)

MULTIPLICIDAD DE FACTORES DE PRODUCTIVIDAD

EVOLUCION TECNOLOGICA DE LA PRODUCTIVIDAD



ESTIMACION

MEDIDA DE PRODUCTIVIDAD EN SENTENCIAS, LOC (LINES OF CODE) O LCF (LINEAS DE CODIGO FUENTE) POR JORNADA, (O DSI: DELIVERED SOURCE INSTRUCTIONS).

NUMERO DE SENTENCIAS POR JORNADA

NUMERO DE SENTENCIAS: TODAS SENTENCIAS FUENTE QUE SE ENTREGAN AL CLIENTE:

- -PROGRAMAS FUENTE CON COMENTARIOS
- -J C L
- -PARAMETROS A PROGRAMAS ESTANDAR (LIBRERIAS, ETC.)

JORNADAS: SE INCLUYEN TODAS LAS CONSUMIDAS EN EL PROYECTO, DESDE EL DISEÑO Y ANALISIS A LA ACEPTACION, TANTO CONSUMIDAS POR EL CLIENTE COMO POR EL PROVEEDOR Y COMPRENDIENDO JEFATURAS, ADMINISTRACION, TIEMPOS NO PRODUCTIVOS, ETC, QUE SE PUEDAN IMPUTAR CLARAMENTE AL PROYECTO. NO SE INCLUYE ACTIVIDAD COMERCIAL, SI LA HUBIERA.

EL NUMERO DE SENTENCIAS DE UN SISTEMA SE CONVIERTE EN EL PARAMETRO QUE DA EL TAMAÑO DEL SISTEMA.

ECONOMIA	CORRECTITUD	ELASTICIDAD
INTEGRIDAD	FIABILIDAD	UTILIZABILIDAD
DOCUMENTACION	MODIFICABILIDAD	CLARIDAD
COMPRENSIBILIDAD	VALIDEZ	MANTENIBILIDAD
FLEXIBILIDAD	GENERALIDAD	PORTABILIDAD
INTEROPERABILIDAD	COMPROBABILIDAD	EFICIENCIA
MODULARIDAD	REUTILIZABILIDAD	

ALGUNOS CRITERIOS DE CALIDAD DEL SOFTWARE

(BUCKLEY, 1984)

ESTIMACION

ALGO SOBRE ALGUNOS CRITERIOS DE CALIDAD: CORRECTITUD, FIABILIDAD:

MODULOS PROPENSOS A DEFECTOS

. 50% DE ERRORES EN SISTEMA GRANDE SE CONCENTRAN EN 5% DE MODULOS. ALREDEDOR DE 60% DE MODULOS SON CORRECTOS.

. EJEMPLOS: EN OS/360 4% DE MODULOS DEL SISTEMA CONCENTRABAN 38% DE TODOS ERRORES: IMS/360, 31 DE 425 MODULOS AGRUPABAN 57% DE TODOS ERRORES.

. LOS MODULOS PROPENSOS A DEFECTOS CONSTITUYEN LAS ENTIDADES MAS COSTOSAS Y MOLESTAS DE LA PROGRAMACION.

. SE HAN IDENTIFICADO SEIS FACTORES CLAVE EN - LOS MODULOS PROPENSOS.

FACTORES PRODOMINANTES EN LOS MODULOS PROPENSOS

1. VARIANZA HUMANA INDIVIDUAL: OCHO PROGRAMADORES PRODUCEN, CON LAS MISMAS ESPECIFICACIONES, CÓDIGO VARIABLE DE 10 A 1 EN CANTIDAD DE ERRORES.

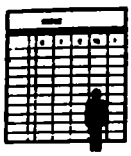
2. INTERACCION DISEÑO-PRUEBAS: ALGUNOS MÓDULOS PROPENSOS NUNCA SON FORMALMENTE VERIFICADOS, PORQUE SE HAN CODIFICADO SIN ESPECIFICACIONES, Y EL PERSONAL QUE LOS PROBÓ NO LAS CONOCIA.

3. CAMBIOS DEL ULTIMO MINUTO EN LOS REQUERIMIENTOS SE HAN ENCONTRADO CON CIERTAS ESTRUCTURAS DE SOFT. DIFÍCILES DE MODIFICAR CON GARANTÍA.

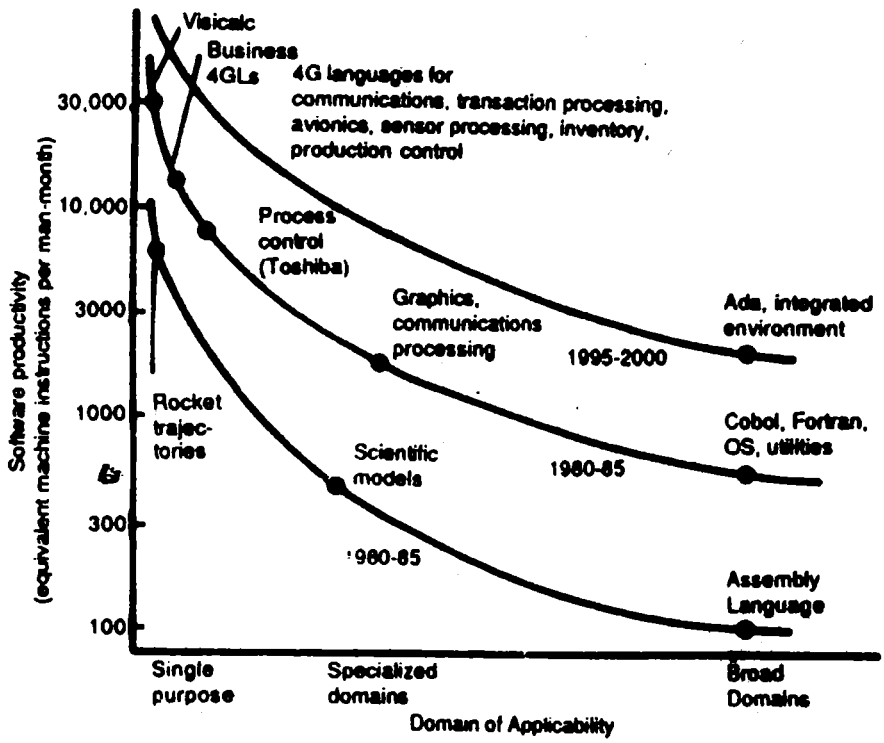
4. CAMBIOS EN EL SOFTWARE SIN REVISAR O PROBAR LAS ACTUALIZACIONES.

5. TAMAÑO DE LOS MODULOS: EL TAMAÑO ÓPTIMO PARECE ESTAR ALREDEDOR DE LAS 50 SENTENCIAS FUENTE. MÓDULOS POR ENCIMA DE 500 TIENDEN A SER DEFECTUOSUS.

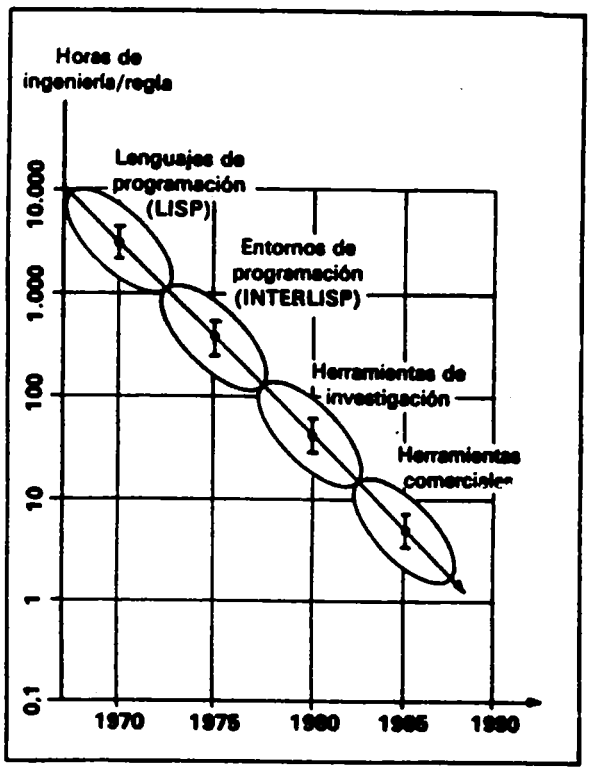
6. COMPLEJIDAD DEL CODIGO EN TERMINOS DE OPERADORES/ /OPERANDOS (METRICA DE HALSTEAD).



ESTIMACION

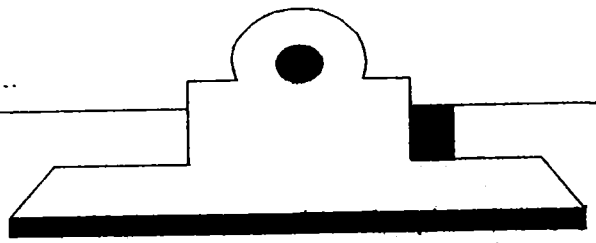


Software technology and productivity trends. (Boehm, 1987)



SIMILITUDES/DIFERENCIAS ENTRE INGENIERIA DEL SOFTWARE E INGENIERIA DEL CONOCIMIENTO

(INFLUENCIA DE HERRAMIENTAS Y ENTORNOS DE PROGRAMACIÓN)

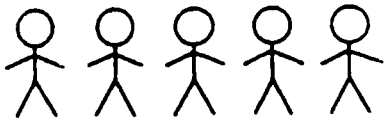


FACTORES HUMANOS SOCIALES

- ☆ EL PROBLEMA DE LA COMUNICACION
- ☆ COMPORTAMIENTO DEL GRUPO PEQUEÑO
- ☆ EQUIPO ESTRUCTURADO DE PROGRAMACION
- ☆ LEYES DE LOS PROYECTOS Y DE LA NATURALEZA HUMANA

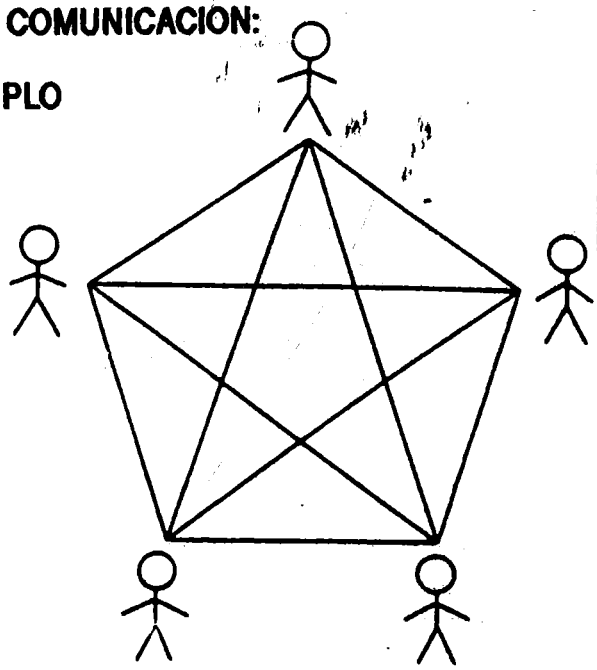
EL PROBLEMA DE LA COMUNICACION:

UN EJEMPLO



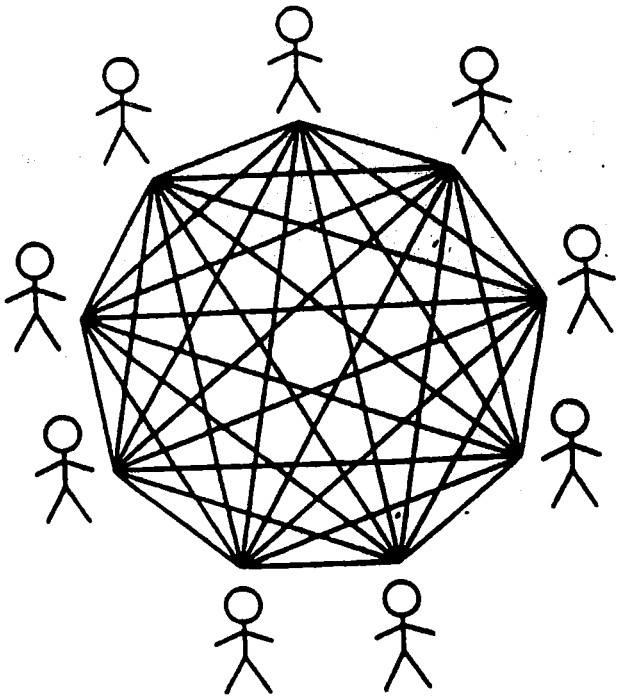
5000 lines/year = 50.000 lines in two years
No communication between programmers

Single projects.



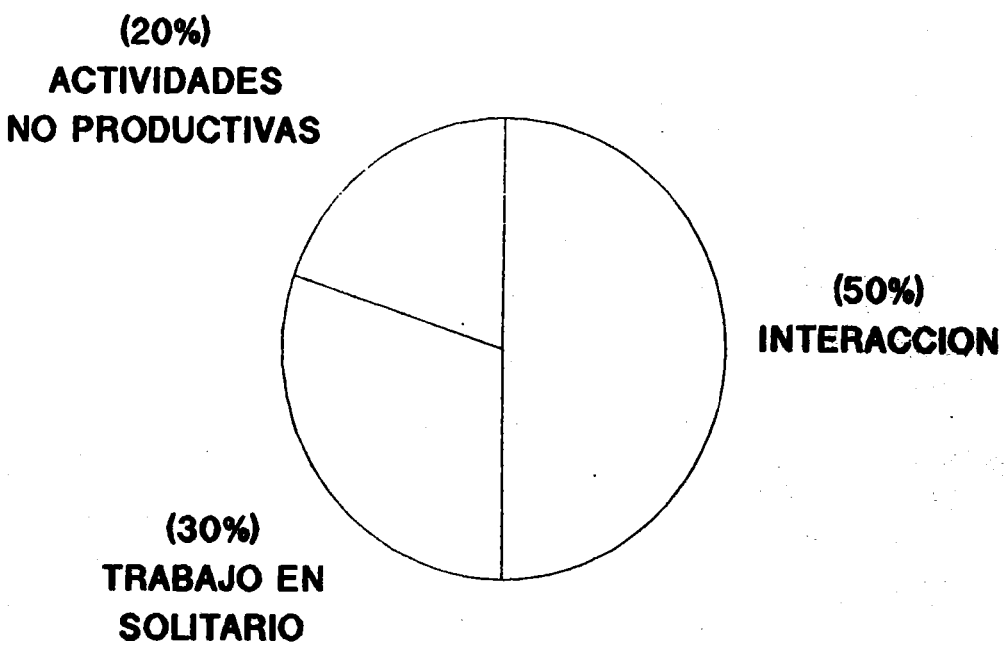
4000 lines/year = 40.000 in two years
Ten communication pairs

Five-member team.

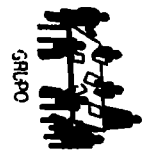


3000 lines/year = 50.000 in two years
Thirty six communication pairs

Nine-member team.



DISTRIBUCION DEL TIEMPO DE UN INGENIERO DE SOFTWARE (SOMMERVILLE, 1988)



-GRUPO PEQUEÑO-

MANDO AUTORITARIO Y COMPORTAMIENTO DE GRUPO

CARACTERISTICAS

ABSOLUTISMO
CENTRALIZADOR DE DECISIONES
FUERTEMENTE DIRECTIVO
ALTAMENTE SUPERVISOR
EVALUACION RIGIDA
NO SE ADMITE LA CRITICA

COMPORTAMIENTO

AGRESIVO, REBELDE
APATICO, SUMISO

RESULTADOS

MEJORA DEL RENDIMIENTO A CORTO PLAZO
FORMACION DE OBJETIVOS PROPIOS DE LOS INDIVIDUOS
DESCENSO DE LA RAPIDEZ Y DE LA CALIDAD
MALESTAR PSIQUICO

-GRUPO PEQUEÑO-

MANDO DEMOCRATICO Y COMPORTAMIENTO DE GRUPO

CARACTERISTICAS

- ADMITE LA DISCUSION
- ELABORACION CONJUNTA DE OBJETIVOS
- PROPORCIONA ORIENTACIONES
- ANIMA AL GRUPO
- COLABORA EN TODOS LOS NIVELES

COMPORTAMIENTO

- PERSONAL Y MAS AMISTOSO
- VALORACION DE LAS CUALIDADES INDIVIDUALES
- SUMISION A LOS INTERESES COMUNES

RESULTADOS

- SE MANTIENE EL NIVEL DE ACTIVIDAD AUNQUE NO ESTE EL JEFE PRESENTE
- CADA MIEMBRO DESEA LA APROBACION DE LOS DEMAS Y POR TANTO LAS DECISIONES SON MAS VINCULANTES
- SE FAVORECE EL DESARROLLO PERSONAL

-GRUPO PEQUEÑO-

MANDO LAISSEZ-FAIRE Y COMPORTAMIENTO DE GRUPO

CARACTERISTICAS

DESAPARECE EL JEFE
O SE TRATA DE UN JEFE BLANDO
CADA SUJETO ES DUEÑO DE SUS ACTOS
NO EXISTE LA VALORACION

COMPORTAMIENTO

MEJORA DE LA CREATIVIDAD
DESAPARICION DE LA NORMA ORIENTATIVA
SE DESARROLLA INDIVIDUALISMO
FALTA DE RESPONSABILIDAD

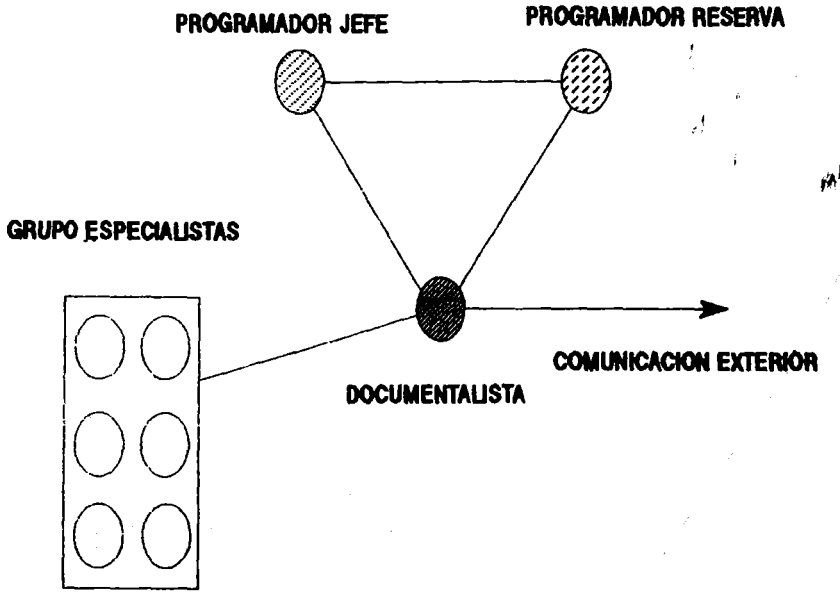
RESULTADOS

SE FAVORECE LA CREATIVIDAD Y LA IMAGINACION
EL TRABAJO NO LLEGA A "ACLARARSE" TOTALMENTE
DESAPARECE LA CONCIENCIA DE GRUPO
SURGE UN LIDER

KEY POINTS

- It is important that software engineers have some understanding of human factors because the ultimate judge of the usefulness of their software are *human* users. If the software does not take their capabilities and limitations into account it will be found wanting.
- Human memory organization is structured into fast, short-term memory, working memory and long-term memory. Mistakes are minimized when transfers between these memory areas are minimized.
- Knowledge falls into two classes, namely arbitrary syntactic knowledge and deeper semantic knowledge. Semantic knowledge is held in some internal way rather than in a language-oriented way.
- In group working, it is common for leaders who are technically competent to emerge. The titular group leader may simply be responsible for administrative activities.
- Personalities working in groups fall into three classes, namely task-oriented, self-oriented and interaction-oriented.
- Democratic groups tend to be more successful than autocratic groups.
- Group interaction should be structured so that the number of group communication links is minimized.
- The workplace has important but unquantifiable effects on software productivity.

(SOMMERVILLE, 1988)



(1) A project administrator who relieves the chief programmer of administrative tasks.

(2) A toolsmith who is responsible for producing software tools to support the project.

(3) A documentation editor who takes the project documentation written by the chief programmer and backup programmer and prepares it for publication.

(4) A language/system expert who is familiar with the idiosyncrasies of the programming language and system which is being used and whose role is to advise the chief programmer on how to make use of these facilities.

(5) A tester whose task is to develop objective test cases to validate the work of the chief programmer.

(6) One or more support programmers who undertake coding from a design specified by the chief programmer. These support programmers are necessary when the scale of the project is such that detailed programming work cannot be carried out by the chief programmer and backup programmer alone.

EL EQUIPO ESTRUCTURADO DE PROGRAMACION (CPT)

(SOMMERVILLE, 1988)

LEYES DE LOS PROYECTOS Y DE LA NATURALEZA HUMANA (EN RELACION CON LOS PROYECTOS).

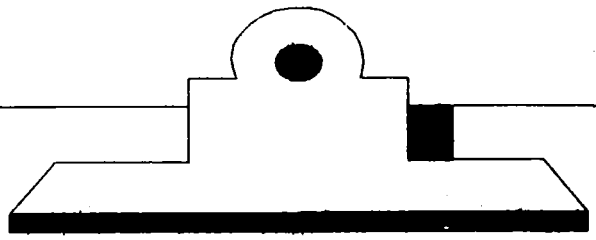
LEY DE PARKINSON. EL TRABAJO TIENDE A OCUPAR TODO EL TIEMPO DISPONIBLE PARA EL (LEY DEL MINIMO ESFUERZO, O DE LA VOLUPTUOSIDAD DE LOS TRABAJOS).

LEY DE BROOKS. AÑADIR GENTE A UN PROYECTO RETRASADO LO ATRASARA AUN MAS (LEY ESTADISTICA DE LOS PROYECTOS).

PRINCIPIO DE PETER. TODA PERSONA TIENDE A ASCENDER HASTA SU NIVEL DE INCOMPETENCIA. (LEY DE LAS ORGANIZACIONES HUMANAS, APLICABLE POR TANTO A JEFES DE PROYECTOS).

LEY DE MURPHY. LAS COSAS PUEDEN EMPEORAR MAS ALLA DE TODO LIMITE (LEY DE PESIMISMO INTEGRAL O DE LA NEGRURA GENERALIZADA).

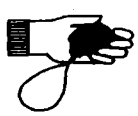
PRINCIPIO DE CLAUSEWITZ. EL MOVIMIENTO DE UN GRUPO DE PERSONAS SE RIGE POR LA VELOCIDAD DE LA MAS LENTA.



MACROESTIMACION: ESTUDIOS SOBRE FACTORES DE PRODUCTIVIDAD

* PREDICTOR DE PRODUCTIVIDAD DE FELIX/WALSTON/IBM

* RANGOS DE PRODUCTIVIDAD Y ARBOL DE MEJORA DE PRODUCTIVIDAD: BOEHM/TRW/ COCOMO.



MACROESTIMACION

METODO DE FELIX/WALTSON/IBM

BASE DE DATOS DE PROYECTOS Y PREDICTOR DE PRODUCTIVIDAD

60 PROYECTOS COMPLETOS

↓

BASE DE DATOS

CALCULOS { PRODUCTIVIDAD
NUMERO MEDIO PERSONAS

ANALISIS REGRESION

↓

29 FACTORES DE PRODUCTIVIDAD

↓

PREDICTOR

"INDICE DE PRODUCTIVIDAD"

← DATOS ENCUESTA NUEVO PROYECTO

↓

PRODUCTIVIDAD ESPERADA

→ ORIENTACION PARA ESTUDIOS MAS DETALLADOS DEL NUEVO PROYECTO.

¿DURACION?

¿TAMAÑO DEL SISTEMA?

¿CANTIDAD Y CLASE DE RECURSOS?

¿DISTRIBUCION TEMPORAL DE RECURSOS?

MACROESTIMACION

EJEMPLO DE METODO DE ESTIMACION

29 VARIABLES QUE MUESTRAN ALTA CORRELACION CON LA PRODUCTIVIDAD.
PROYECTO IBM

CUESTIÓN O VARIABLE	RESPUESTA GRUPAL PRODUCTIVIDAD MEDIA			CAMBIO DE PRODUCTIVIDAD
				DSI/MM
COMPLEJIDAD RELACIÓN CON CLIENTE	< NORMAL 500	NORMAL 295	> NORMAL 124	376
PARTICIPACIÓN USUARIO EN DEFINICIÓN REQUERIMIENTOS	NINGUNA 491	ALGUNA 267	MUCHA 205	286
CAMBIOS EN DISEÑO ORIGINADOS EN EL CLIENTE	POCOS 297		MUCHOS 196	101
EXPERIENCIA CLIENTE - CON ÁREA APLICACIÓN PROYECTO	NINGUNA 318	ALGUNA 340	MUCHA 206	112
EXPERIENCIA GENERAL Y CAPACITACIÓN PERSONAL	BAJOS 132	MEDIOS 257	ALTOS 410	278
PORCENTAJE PROGRAMADORES EN EL DESARROLLO PARTICIPARON EN DISEÑO ESPECIFICACIONES - FUNCIONALES.	< 25% 153	25-50% 242	>50% 391	238
EXPERIENCIA PREVIA - CON ORDENADOR PROYECTO	MÍNIMA 146	MEDIA 270	EXTENSA 312	166
EXPERIENCIA PREVIA CON LENG. PROGRAMACIÓN	MÍNIMA 122	MEDIA 225	EXTENSA 385	263
EXPERIENCIA PREVIA CON APLICACIÓN DE SIMILAR O MAYOR TAMAÑO Y COMPLEJIDAD	MÍNIMA 146	MEDIA 221	EXTENSA 410	264
RELACIÓN DE Nº MEDIO PERSONAS A DURACIÓN (PERSONAS/MES)	< 0.5 305	0.5-0.9 310	> 0.9 173	132

MACROESTIMACION

CUESTIÓN O VARIABLE	RESPUESTA GRUPAL PRODUCTIVIDAD MEDIA			CAMBIO DE PRODUCTIVIDAD
HARDWARE DESARROLLÁNDOSE EN PARALELO	NO 297		SI 177	120
ACCESO AL COMPUTADOR DURANTE DESARROLLO, ABIERTO CON PETICIÓN ESPECIAL	0% 226	1-25% 274	> 25% 357	131
ACCESO AL COMPUTADOR CERRADO	0-10% 303	11-85% 251	> 85% 170	133
ENTORNO BAJO CONDICIONES DE SEGURIDAD DEL COMPUTADOR Y 25% DE PROGRAMAS Y DATOS	NO 289		SI 156	133
PROGRAMACIÓN ESTRUCTURADA	0-33% 169	34-66%	>66% 301	132
INSPECCIÓN DISEÑO Y CODIFICACIÓN	0-33% 220	34-66% 300	>66% 339	119
DESARROLLO DESCENDENTE	0-33% 196	34-66% 237	>66% 321	125
EMPLEO DEL EQUIPO ESTRUCTURADO DE PROGR. (C.P.T.)	0-33% 219	34-66% -	>66% 408	189
COMPLEJIDAD GENERAL DEL CÓDIGO DESARROLLADO	<MEDIA 314		>MEDIA 185	129
COMPLEJIDAD DE PROCESAMIENTO APLICACIÓN	<MEDIA 349	MEDIA 345	>MEDIA 168	181
COMPLEJIDAD DEL FLUJO DE PROGRAMAS	<MEDIA 289	MEDIA 299	>MEDIA 209	80
LIMITACIONES GENERALES AL DISEÑO PROGRAMAS	MÍNIMAS 293	MEDIAS 286	SEVERAS 166	107

MACROESTIMACION

CUESTION O VARIABLE	RESPUESTA GRUPAL PRODUCTIVIDAD MEDIA			CAMBIO DE PRODUCTIVIDAD
LÍMITACIONES DISEÑO PROGRAMAS EN MEMO-- RIA PRINCIPAL.	MÍNIMAS 391	MEDIAS 277	SEVERAS 193	198
LIMITACIONES DISEÑO PROGRAMAS EN TIEMPO	MÍNIMAS 303	MEDIAS 317	SEVERAS 171	132
CÓDIGO PARA OPERA-- CIÓN EN TIEMPO REAL O INTERACTIVO O PA- RA EJECUCIÓN BAJO SEVERAS CONDICIONES TEMPORALES.	<10% 279	10-40% 337	>40% 203	76
PORCENTAJE DE CÓDI- GO PARA ENTREGAR	0-90% 159	91-99% 327	100% 265	106
CÓDIGO CLASIFICADO COMO APLICACIÓN NO MATEMÁTICA Y PRO-- GRAMAS FORMATO E/S.	0-33% 188	34-66% 311	67-100% 267	79
Nº DE CLASES DE - ITEMS EN BASE DA-- TOS POR 1000 LOC.	0-15% 334	16-80% 243	>80% 193	141
Nº PÁGINAS DOCUMEN TACIÓN ENTREGADA - POR 1000 LOC ENTRE GADAS	0-32 320	33-88 252	>88 195	125

MACROESTIMACION

INDICE DE PRODUCTIVIDAD: PREDICTOR

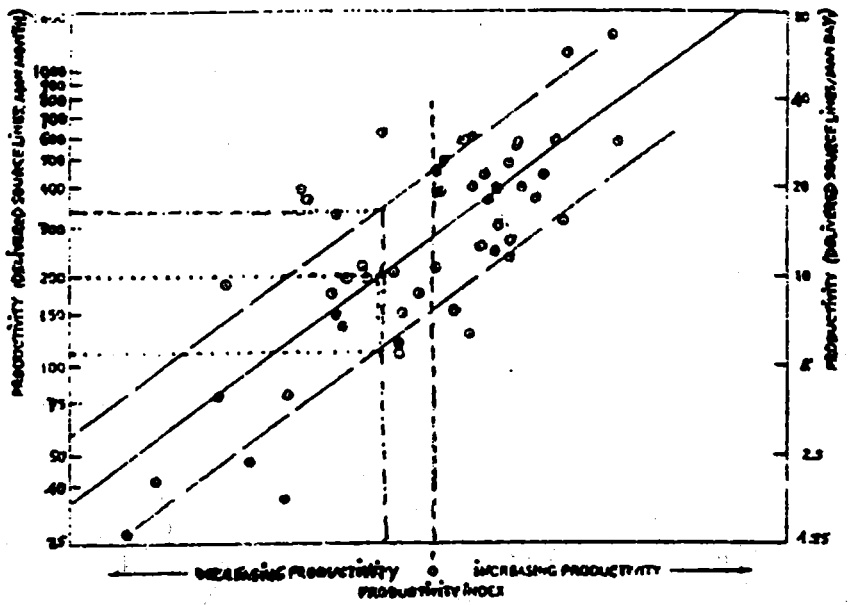
$$I = \sum_{I=1}^{29} W_I X_I$$

I: INDICE DE PRODUCTIVIDAD PARA UN PROYECTO

W_I: PESO DE LA CUESTION (O VARIABLE), CALCULADO POR UN MEDIO DEL LOGARITMO DECIMAL DEL CAMBIO DE PRODUCTIVIDAD CORRESPONDIENTE A ESA CUESTION.

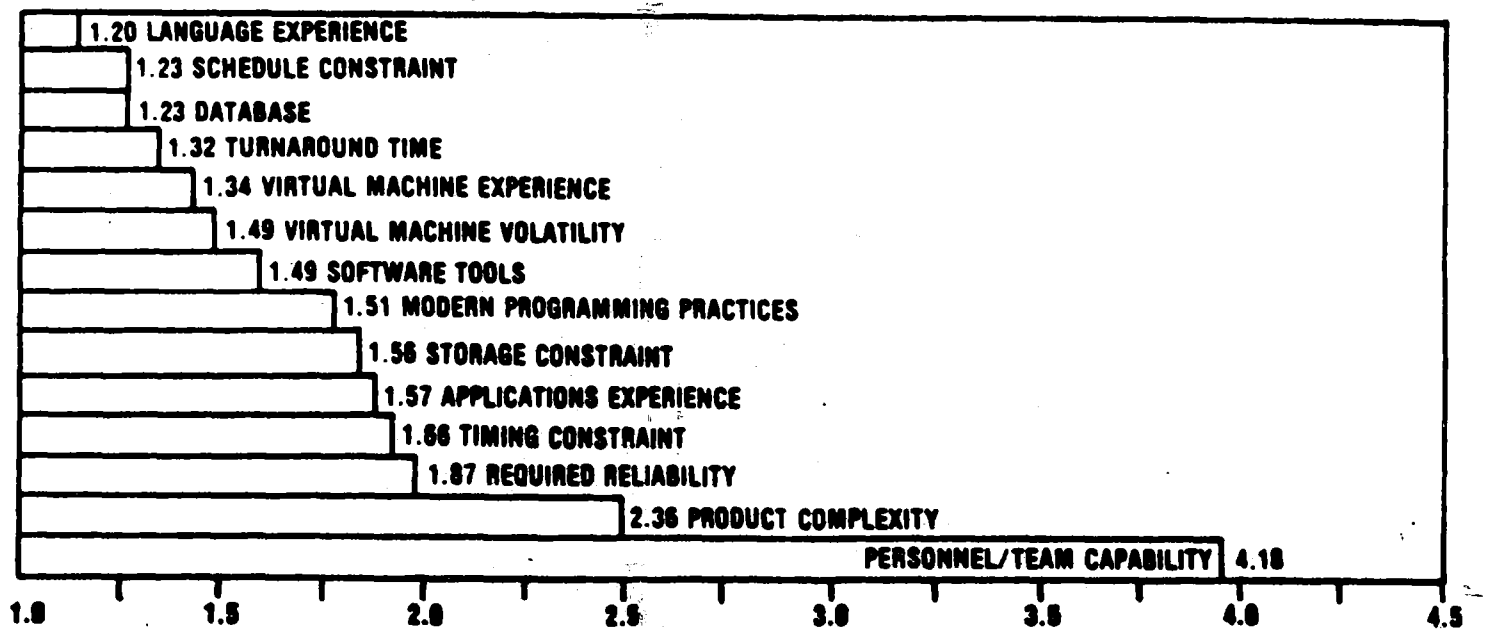
X_I: RESPUESTA A LA CUESTION (+ 1, 0, -1), SEGUN AQUELLA INDIQUE PRODUCTIVIDAD AUMENTADA, NOMINAL O REDUCIDA.

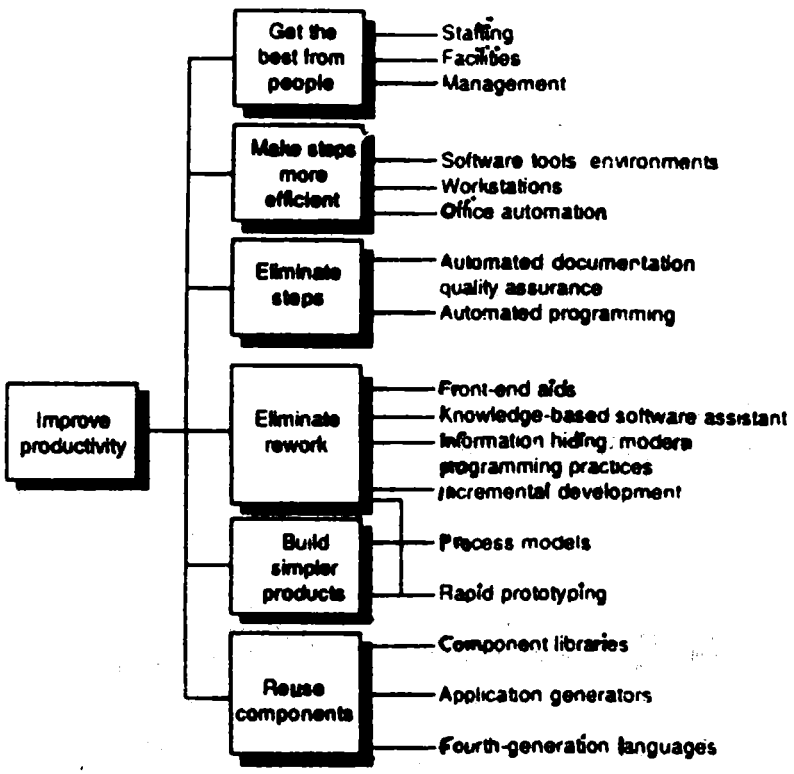
RELACION ENTRE PRODUCTIVIDAD E INDICE DE PRODUCTIVIDAD CALCULADO PARA 29 VARIABLES (DATOS DE 51 PROYECTOS).



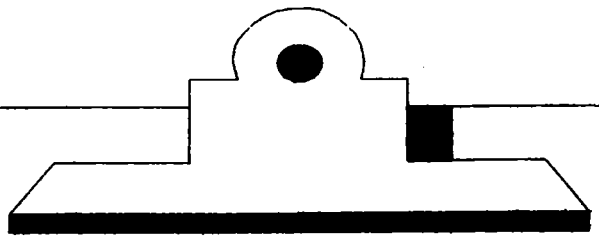
LAS LINEAS DE TRAZOS REPRESENTAN LA DESVIACION ESTANDAR CON RESPECTO A LA LINEA, OBTENIDA POR ANALISIS DE REGRESION CON AJUSTE POR MINIMOS CUADRADOS.

EL EJEMPLO, EN RESPUESTA AL CUESTIONARIO, DIO UN INDICE DE PRODUCTIVIDAD EQUIVALENTE A UNA PRODUCTIVIDAD DE 200 LINEAS DE CODIGO CON UN MARGEN DE ERROR ESTANDAR DE 115 A 340 APROXIMADAMENTE.





Software productivity improvement opportunity tree. (Boehm, 1987)

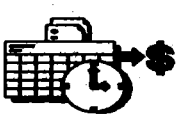


METODOS Y MODELOS DE MACROESTIMACION

* RELACIONES DE WALSTON/FELIX/IBM

* METODO DE PUTNAM

* MODELO COCOMO (TRW/BOEHM)



WALSTON/FELIX/IBM

$E = 5{,}2 \cdot L^{0{,}91}$

$M = 4{,}1 \cdot L^{0{,}36}$

$M = 2{,}47 \cdot E^{0{,}35}$

E: ESFUERZO TOTAL MESES x HOMBRE

L: ($x10^3$) LINEAS DE CODIGO

M: DURACION PROYECTO, MESES

ESTAS RELACIONES NO INCLUYEN ESFUERZOS

DE MANTENIMIENTO

METODO DE PUTNAM

$$\dot{Y} = 2KATE^{-AT^2} = \frac{K}{T_D^2} T.E^{-\frac{T^2}{2T_D^2}}$$

CURVA DE RAYLEIGH-NORDEN DE DISTRIBUCION TEMPORAL DE LA PLANTILLA (MANPOWER)

$$S_S = C_K K^{1/3} T_D^{4/3}$$

ECUACION EMPIRICA DEL SOFTWARE

$\dot{Y}$: NIVEL INSTANTANEO DE PLANTILLA

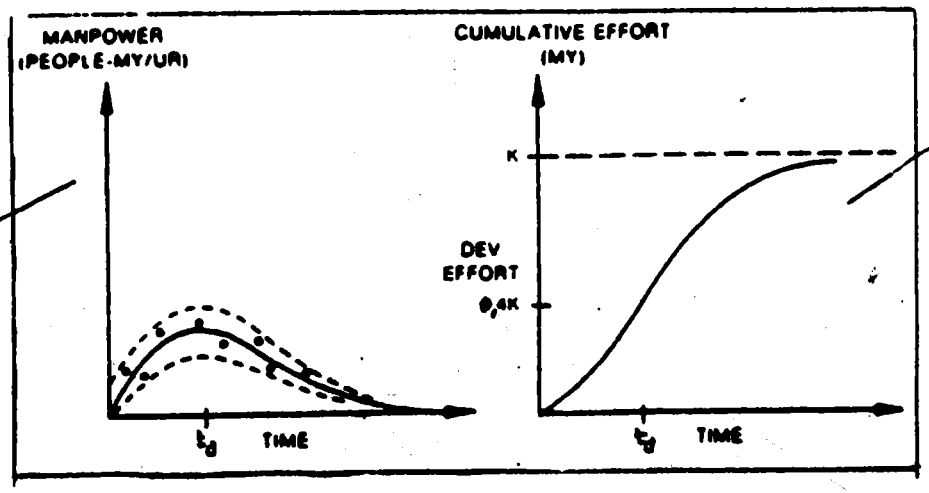
K : ESFUERZO HUMANO TOTAL CICLO VIDA: HOMBRES-AÑO

T_D: TIEMPO PARA $\dot{Y}$ MAXIMO ($\simeq$ TIEMPO DESARROLLO)

S_S: TAMAÑO SOFTWARE EN LINEAS (SENTENCIAS) DE CODIGO FUENTE

C_K: FACTOR TECNOLOGICO O CONSTANTE TECNOLOGICA AMBIENTAL.

$$\dot{Y} = 2KATE^{-AT^2}$$



$$Y = K(1-E^{-AT^2})$$

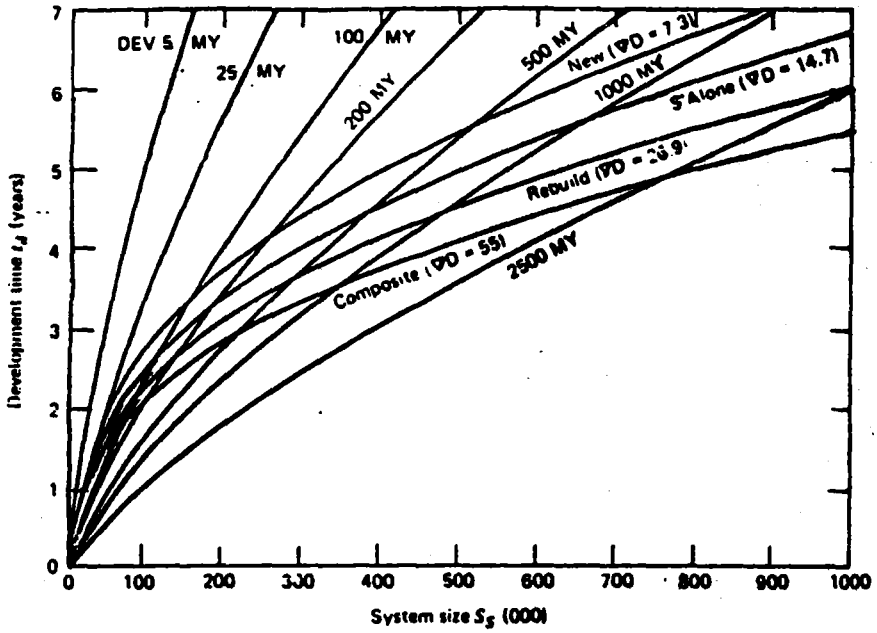
VALIDA PARA SISTEMAS GRANDES

ESFUERZO DE DESARROLLO
(E ó DE ó DEV)
E = 0,4 K.

ECUACION DE SOFTWARE

METODOS Y MODELOS

Size–Time–Effort Trade-Off Chart: $S_s = C_k K^{1/3} t_d^{4/3}$, $C_k = 4984$.



$$S_S = C_K K^{1/3} T_D^{4/3}$$

C_K: EL ESTADO DE LA TECNOLOGIA APLICADA

EL ENTORNO TECNICO Y ORGANIZATIVO PARA EL DESARROLLO

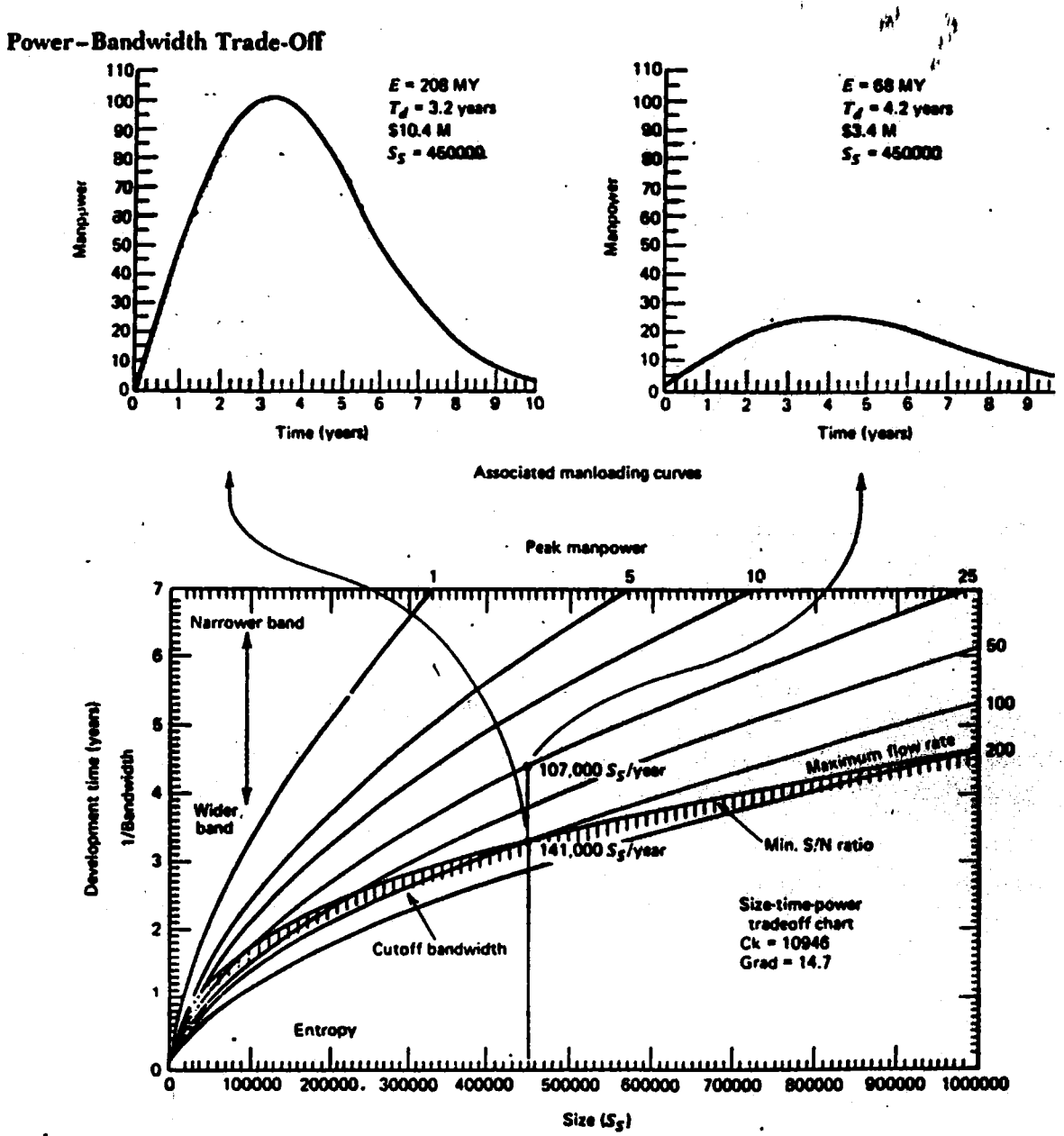
EL EQUIPO DE DESARROLLO DISPONIBLE Y EL TIEMPO NECESITADO PARA DEPURAR Y PROBAR

LA INTENSIDAD DE UTILIZACION DE TECNICAS MODERNAS DE PROGRAMACION

$D = \frac{K}{T_D^2}$ (PARAMETRO DE DIFICULTAD)

$\nabla D = \frac{K}{T_D^3}$ (GRADIENTE DE LA DIFICULTAD)

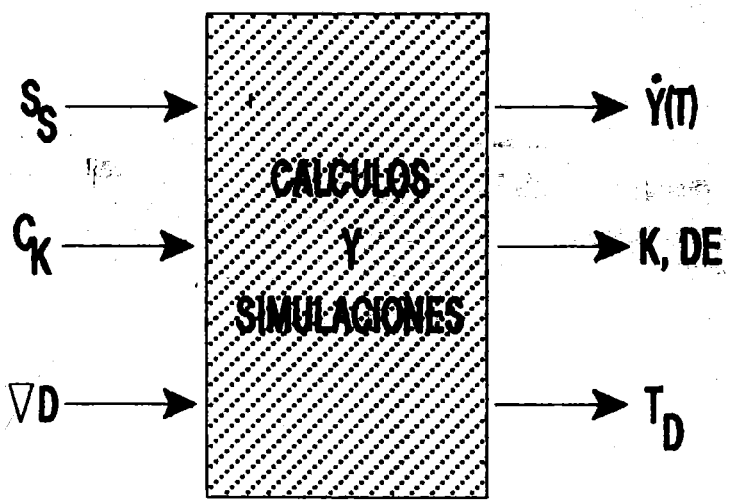
∇ D TOMA VALORES DISCRETOS: 8 SI EL SISTEMA ES ENTERAMENTE NUEVO Y TIENE MUCHAS INTERFACES Y RELACIONES CON OTROS SISTEMAS, 15 SI ES AUTONOMO Y NUEVO, 27 SI ES UNA RECONSTRUCCION CON MUCHA LOGICA Y CODIGO PREEXISTENTES, ETC...



EFECTO DE LOS CAMBIOS IMPUESTOS SOBRE T_D EN EL COSTE Y EN LA PRODUCTIVIDAD.

METODOS y MODELOS

METODOLOGIA PRACTICA DE ESTIMACION DE PUTNAM



METODOS Y MODELOS

PROCEDIMENTO DE MACROESTIMACION DE PUTNAM, SOBRE LA CURVA DE RAYLEIGH.

1. ESTIMACION TAMAÑO SISTEMA, EN SENTENCIAS FUENTE. BASADA EN EXPERIENCIA Y APROXIMACIONES SUCESIVAS.

S_S

2. CURVA DE TECNOLOGIA, RELACION EMPIRICA SEGUN ESTANDARES DE LA INSTALACION.

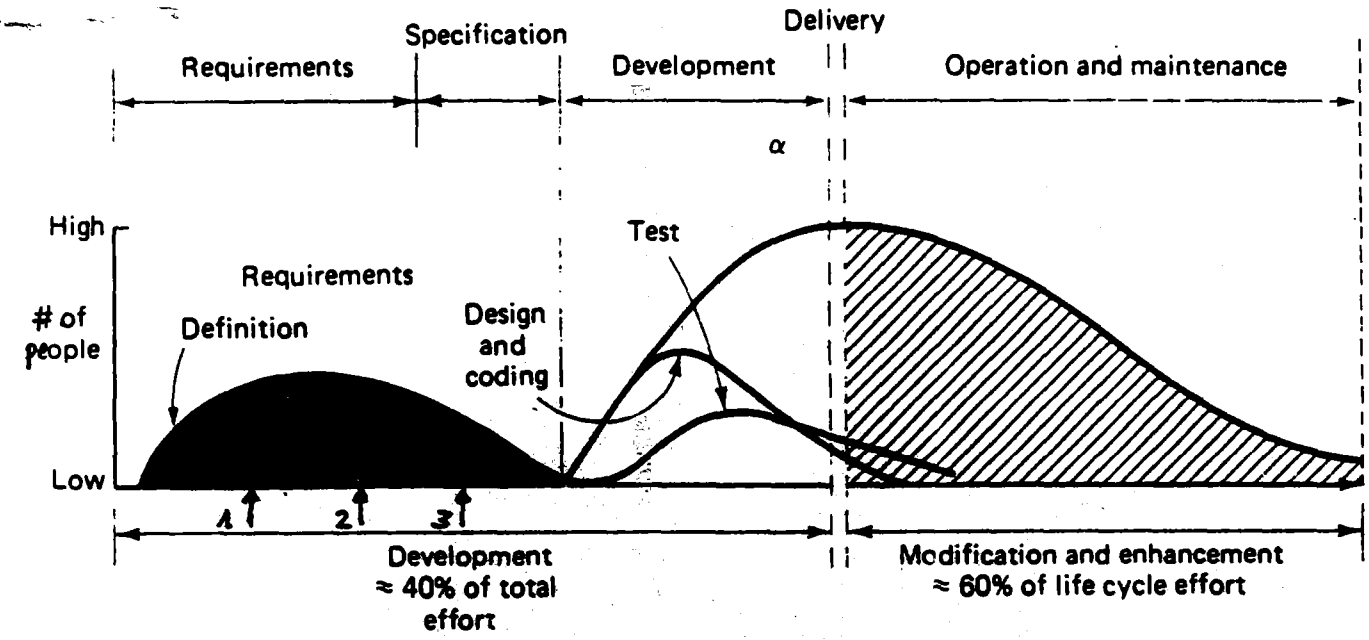
K (O DE, T_D) DEVELOPMENT EFFORT = 0,4.K)

3. ANALISIS DE RIESGOS, SIMULACION DE INCERTIDUMBRES Y AJUSTES FINALES PARA CONSIDERAR RESTRICCIONES DE ORDEN SUPERIOR AL PROYECTO.

CURVA DE PLANIFICACION GLOBAL

MACROESTIMACION

PUTNAM. 1. ESTIMACION TAMAÑO SISTEMA.



METODOS Y MODELOS

EJEMPLO. PROYECTO SAVE: SISTEMA DE CONTROL DE INVENTARIOS

PUNTO 1 DEL CICLO

MÁX. NUM. SENTENCIAS FUENTE B = 140.000

MIN. NUM. SENTENCIAS FUENTE A = 50.000

NUM. PROMEDIO $\frac{A+B}{2}$ = 95.000

DESVIACIÓN TÍPICA = $\frac{|B-A|}{6}$ = 15.000

TAMAÑO ESPERADO = 95.000 ± 15.000

HIPÓTESIS GAUSS. ESTO QUIERE DECIR QUE SE SUPONE QUE EL TAMAÑO DEL SOFT. TENDRÁ, CON PROBABILIDAD 68%, ENTRE 80.000 Y 110.000 SENTENCIAS.

PUNTO 2 DEL CICLO

YA HAY UN DESPIECE EN GRANDES SUBSISTEMAS, CON UN PROMEDIO DE LAS OPINIONES DELPHI DE LOS EXPERTOS ASÍ:

FUNCIÓN	MAS PEQUEÑO A	MAS PROBABLE M	MAYOR B	ESPERADO	DESVIACION TIP.
MANEJADORES FICHEROS	25.000	40.000	70.000	42.500	7.500
UTILIDADES	5.000	15.000	26.000	15.167	3.500
PROCEDIMIENTOS DEL SISTEMA	12.000	36.000	50.000	34.333	6.333
				92.000	10.422

HIPOTESIS DISTRIBUCION BETA

VALOR ESPERADO = $\frac{A+4M+B}{6}$

DESVIACIÓN PARA CADA SUBSISTEMA $\frac{|B-A|}{6}$

DESVIACIÓN TOTAL = $\sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}$

PUNTO 3 DEL CICLO

DESPIECE A MAYOR DETALLE, YA SUPERADO EL PICO DE ESFUERZO DE ELABORACIÓN DE ESPECIFICACIONES DE DISEÑO FUNCIONAL.

DIMENSIONAMIENTO

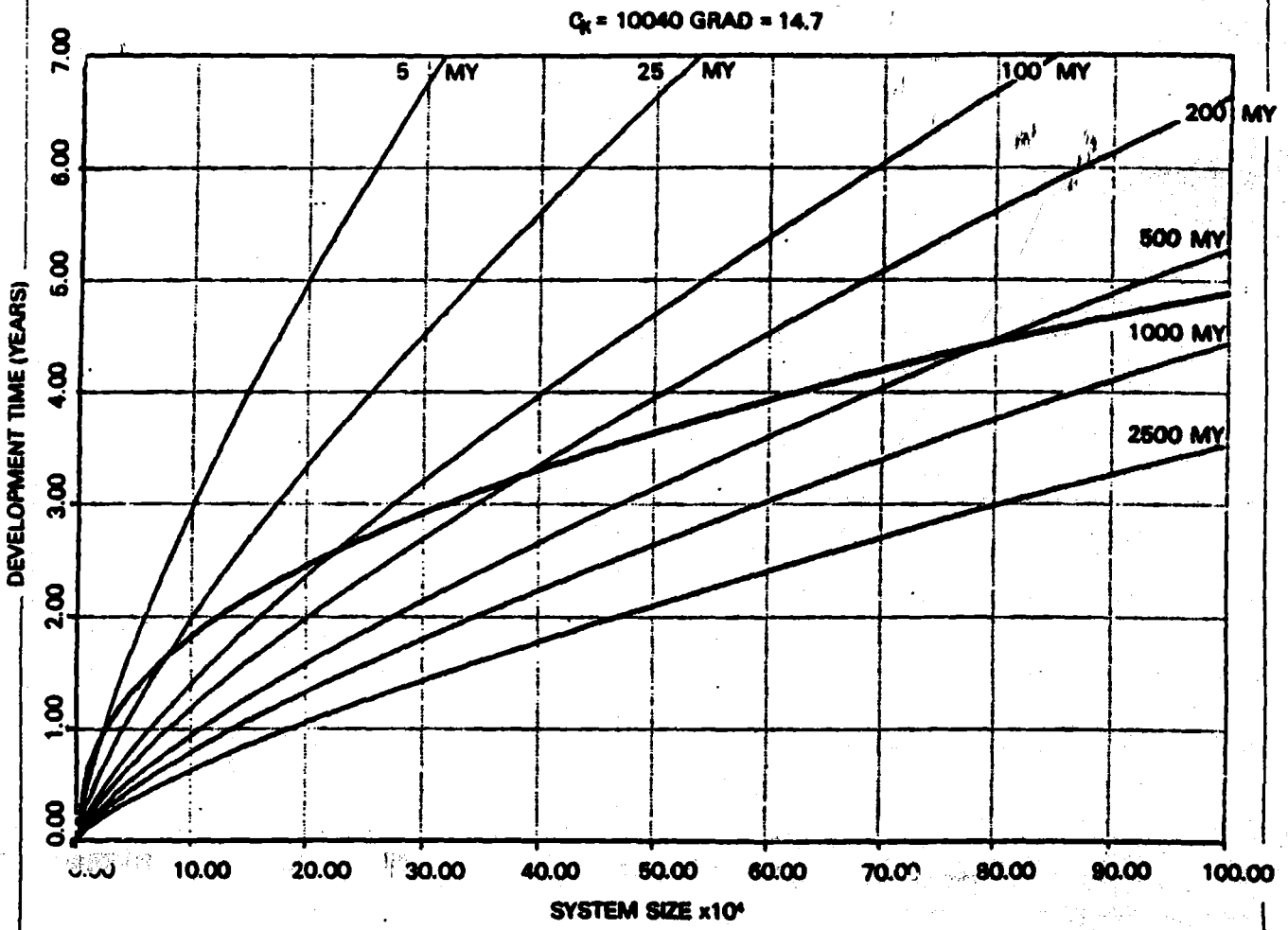
TÍTULO: SAVE (DISEÑO FUNCIONAL) **FECHA**

FUNCIÓN	MÁS PEQUEÑO	MÁS PROBABLE	MAYOR	ESPERADO	DESV.TÍPICA
A1	8675.	13375.	18625.	13467.	1658.
A2	5577.	8988.	13125.	9109.	1258.
A3	3160.	3892.	8800.	4588.	940.
A4	850.	1425.	2925.	1579.	346.
A5	1875.	4052.	8250.	4389.	1063.
A6	1437.	2455.	6125.	2897.	781.
A7	6875.	10625.	16250.	10938.	1563.
A8	5830.	8962.	17750.	9905.	1987.
A9	9375.	14625.	28000.	15979.	3104.
A10	6300.	13700.	36250.	16225.	4992.
A11	5875.	8975.	14625.	9400.	1458.
TOTAL				98475.	7081.

CON PROBABILIDAD 68%, EL SISTEMA TENDRA DE 91.394 A 105.556

CON PROBABILIDAD 99%, EL SISTEMA TENDRA DE 77.231 A 119.718

METODOS Y MODELOS



Size-effort-time trade-off chart

		t_d = 2 years	Fastest		Risk Biased	
	S_s	Dev Effort (MY)	t_d	Dev Effort (MY)	t_d + .4 yr	Dev Effort (MY)
-3σ	77,000	11.28 ($.564M)	1.63	25.80 ($1.29M)	2.03	10.71 ($.55M)
-1σ	91,394	18.86 ($.943)	1.75	32.16 ($1.61)	2.15	14.12 ($.71M)
E	98,475	23.59 ($1.180M)	1.81	35.40 ($1.77M)	2.21	15.91 ($.796M)
+1σ	105,556	29.05 ($1.45M)	1.86	38.71 ($1.84M)	2.26	17.77 ($.89M)
+3σ	120,000	42.69 ($2.135M)	1.97	45.65 ($2.28M)	2.37	21.77 ($1.09M)

Assumption: On-line, interactive development, Top down structured programming, HOL, Contemporary development environment. No machine constraints.
C_k = 10040; Standalone System—| ∇D | = 15

CONTINGENCIAS EXTERIORES A LA GESTION

Nº MAX. DE PERSONAS EN PICO ESFUERZO	28
Nº MIN. " " " "	15
COSTE MAXIMO	$ 2M
T_D MAXIMO	2 AÑOS

SOLUCIONES POR ⇓ PROG. LINEAL

	TIEMPO MINIMO	COSTE MINIMO
T_D(AÑOS)	1.83	2
ESFUERZO DESARROLLO	33.6 H x A	24.4 H x A
COSTE	$ 1.68 M	$ 1.22 M

COMPLETAR RANGO POSIBILID. ⇓ CURVAS DESARROLLO SOFT.

T_D	HOMB x AÑO	COSTE	
1.83	33.6	$ 1.68 M	
1.87	30.3	1.52	
1.91	27.8	1.39	RIESGO ACEPTABLE
1.95	25.5	1.28	
2.00	24.4	1.22	MAXIMO RIESGO DE NO ESTAR EN PLAZO

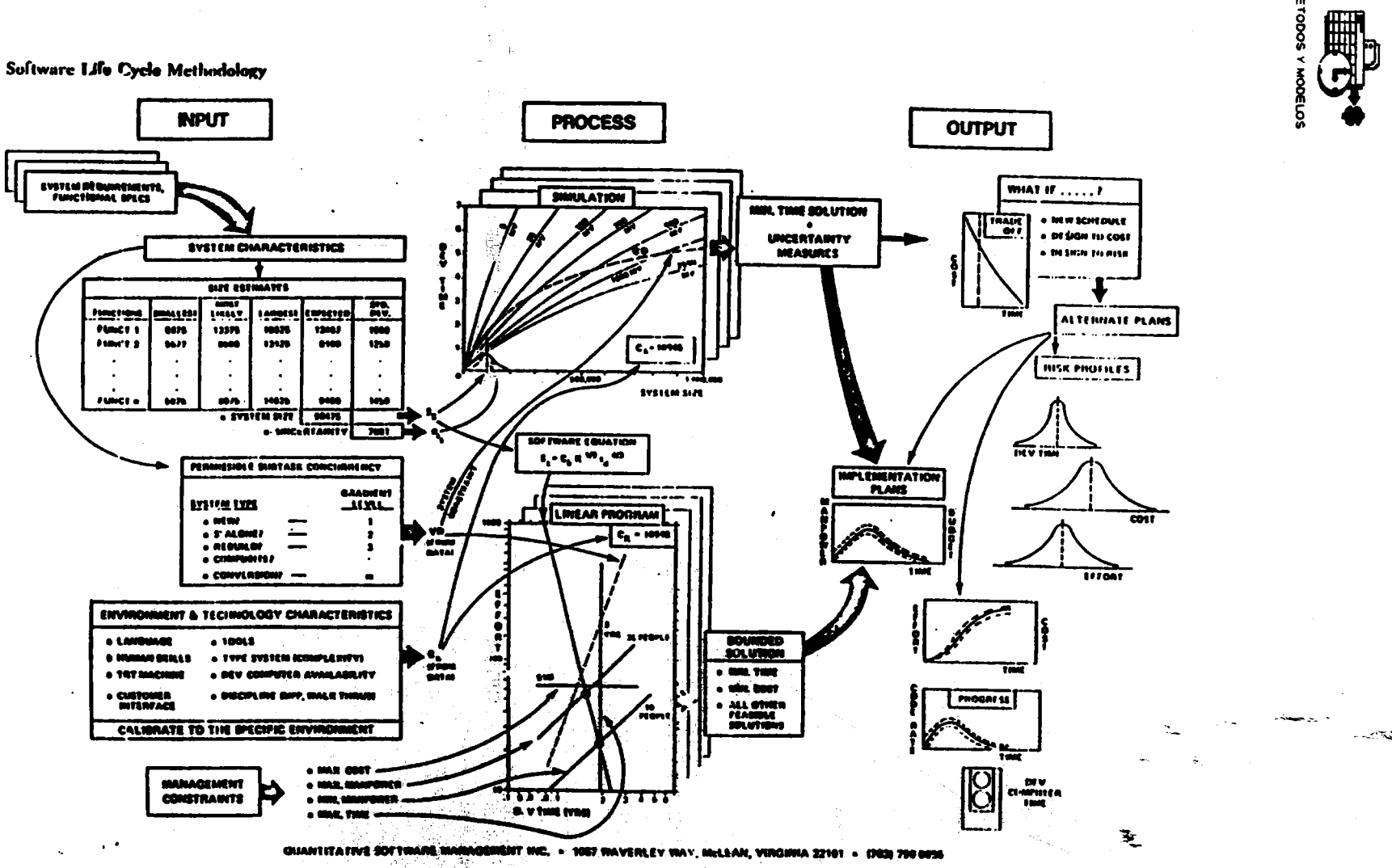


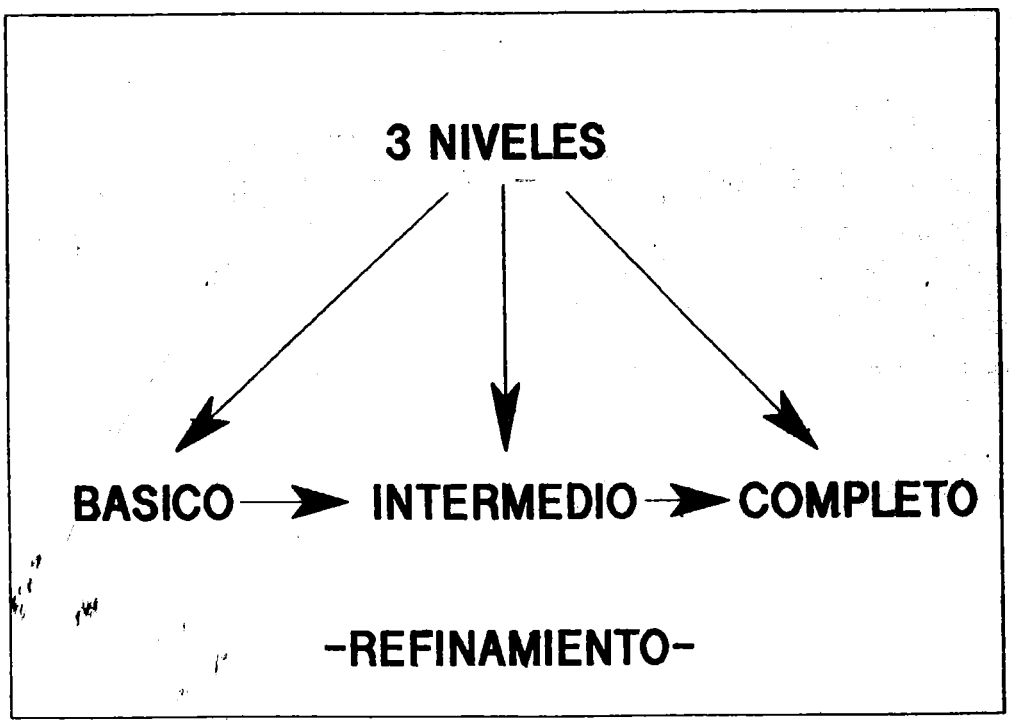
GRAFICO DE LA METODOLOGIA DE PUTNAM COMPLETA

METODOS Y MODELOS

MODELO COCOMO

(TRW, BOEHM)

(CONSTRUCTIVE COST MODEL)



METODOS Y MODELOS

COCOMO BASICO

* TRES GRADOS DE DIFICULTAD → TAMAÑO
                            → TIPO

* ALGORITMO:

TAMAÑO SOFTWARE
↓
ESFUERZO (COSTE)
↓
TIEMPO DE DESARROLLO

* $PM = 2,4 \cdot (KDSI)^{1,05} \longrightarrow TDEV = 2,5 \cdot (PM)^{0,38}$

** $PM = 3 \cdot (KDSI)^{1,12} \longrightarrow TDEV = 2,5 \cdot (PM)^{0,35}$

*** $PM = 3,6 \cdot (KDSI)^{1,20} \longrightarrow TDEV = 2,5 \cdot (PM)^{0,32}$

KDSI: MILES DE INSTRUCCIONES-FUENTE ENTREGADAS (SIN COMENTARIOS).

PM: ESFUERZO, PERSONAS-MES

TDEV: TIEMPO DE DESARROLLO, EN MESES

(1 PM = 152 HORAS DE TRABAJO REAL/MES)

(PRODUCTIVIDAD 1^{ER}. GRADO: 16 DSI/PERSONA-DIA)

(PRODUCTIVIDAD 3^{ER}. GRADO: 4 DSI/PERSONA-DIA)

Project attribute multipliers

Cost driver	*Ratings*					
	VL	*L*	*N*	*H*	*VH*	*EH*
RELY	0.75	0.88	1.00	1.15	1.40	—
DATA	—	0.94	1.00	1.08	1.16	—
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME	—	—	1.00	1.11	1.30	1.66
STOR	—	—	1.00	1.06	1.21	1.56
VIRT	—	0.87	1.00	1.15	1.30	—
TURN	—	0.87	1.00	1.07	1.15	—
ACAP	1.46	1.19	1.00	0.86	0.71	—
AEXP	1.29	1.13	1.00	0.91	0.82	—
PCAP	1.42	1.17	1.00	0.86	0.70	—
VEXP	1.21	1.10	1.00	0.90	—	—
LEXP	1.14	1.07	1.00	0.95	—	—
MODP	1.24	1.10	1.00	0.91	0.82	—
TOOL	1.24	1.10	1.00	0.91	0.83	—
SCED	1.23	1.08	1.00	1.04	1.10	—

Tables 11.1 and 11.2 are reproduced from *Software Engineering Economics*, B.W. Boehm (1981), by permissi·n of Prentice-Hall Inc.

FACTORES DE COSTE VERSUS MULTIPLICADORES

$$\text{APLICACION: PM} \times \prod_{1}^{15} \text{(MULTIPLICADORES)}$$

METODOS Y MODELOS

COCOMO INTERMEDIO

* ALGORITMO: PM_{BASICO} x FACTORES CORRECTIVOS

* ATRIBUTOS (DRIVERS) ——> FACTORES CORRECTIVOS (MULTIPLICADORES)

ATRIBUTOS DEL PRODUCTO

- RELY (REQUIRED SOFTWARE RELIABILITY)
- DATA (DATABASE SIZE)
- CPLX (PRODUCT COMPLEXITY)

ATRIBUTOS DEL COMPUTADOR

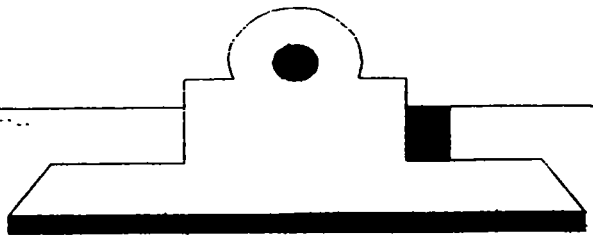
- TIME (EXECUTION TIME CONSTRAINTS)
- STOR (STORAGE CONSTRAINTS)
- VIRT (VIRTUAL MACHINE VOLATILITY)
- TURN (COMPUTER TURNAROUND TIME)

ATRIBUTOS DEL PERSONAL

- ACAP (ANALYST CAPABILITY)
- AEXP (APPLICATION EXPERIENCE)
- VEXP (VIRTUAL MACHINE EXPERIENCE)
- PCAP (PROGRAMMER CAPABILITY)
- LEXP (PROGRAMMING LANGUAGE EXPERIENCE)

ATRIBUTOS DEL PROYECTO

- MODP (MODERN PROGRAMMING PRACTICES)
- TOOL (SOFTWARE TOOLS)
- SCED (REQUIRED DEVELOPMENT SCHEDULE)



MANTENIMIENTO DEL SOFTWARE

* FACTORES DE COSTE
* CATEGORIAS DE MANTENIMIENTO
* COCOMO, DE NUEVO
* RELACION CON GESTION DE CONFIGURACIONES Y CONTROL DE CALIDAD

FACTORES NO TECNICOS DE COSTE

- ☆ GRADO DE DEFINICION DE LA APLICACION ↘
- ☆ ESTABILIDAD EQUIPO HUMANO ↘
- ☆ EDAD DEL PROGRAMA ↗
- ☆ DEPENDENCIA RESPECTO ENTORNO EXTERIOR ↗
- ☆ ESTABILIDAD HARDWARE ↘

FACTORES TECNICOS

- ☆ INDEPENDENCIA ENTRE MODULOS ↘
- ☆ NIVEL LENGUAJE ↘
- ☆ PERFECCION ESTILO PROGRAMACION ↘
- ☆ TIEMPO DEDICADO A VALIDACION Y PRUEBAS ↘
- ☆ CANTIDAD Y CALIDAD DOCUMENTACION PROGRAMAS ↘

CAMBIOS

PERFECTIVO **(funcionalidad)**	**65%**
ADAPTATIVO **(entorno)**	**18%**
CORRECTIVO **(errores)**	**17%**

DISTRIBUCION DE LOS COSTES DE MANTENIMIENTO
(Lientz, Swanson, 1980)

CATEGORIAS DE MANTENIMIENTO

MANTENIMIENTO

C O C O M O

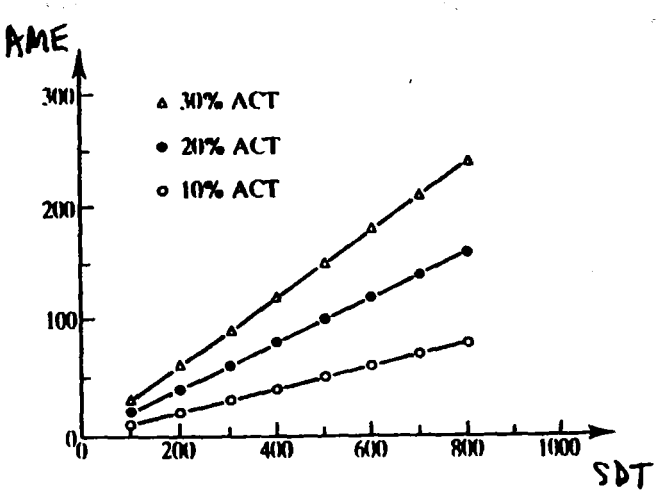
PARA MANTENIMIENTO

$$AME = 1{,}0 \cdot ACT \cdot SDT$$

ANNUAL MAINTENANCE EFFORT (PERSONAS-MES)

ANNUAL CHANGE TRAFFIC (%)

SOFTWARE DEVELOPMENT TIME (PERSONAS-MES)



ACT: FRACCION DE LAS INSTRUCCIONES-FUENTE QUE CAMBIARAN DURANTE UN AÑO, POR ADICION O MODIFICACION

- FACTORES DE COSTE VERSUS MULTIPLICADORES -

APLICACION : AME $\times \prod_{1}^{15}$ (MULTIPLICADORES)

¡OJO¡ : CAMBIOS CON RESPECTO COCOMO PARA DESARROLLO

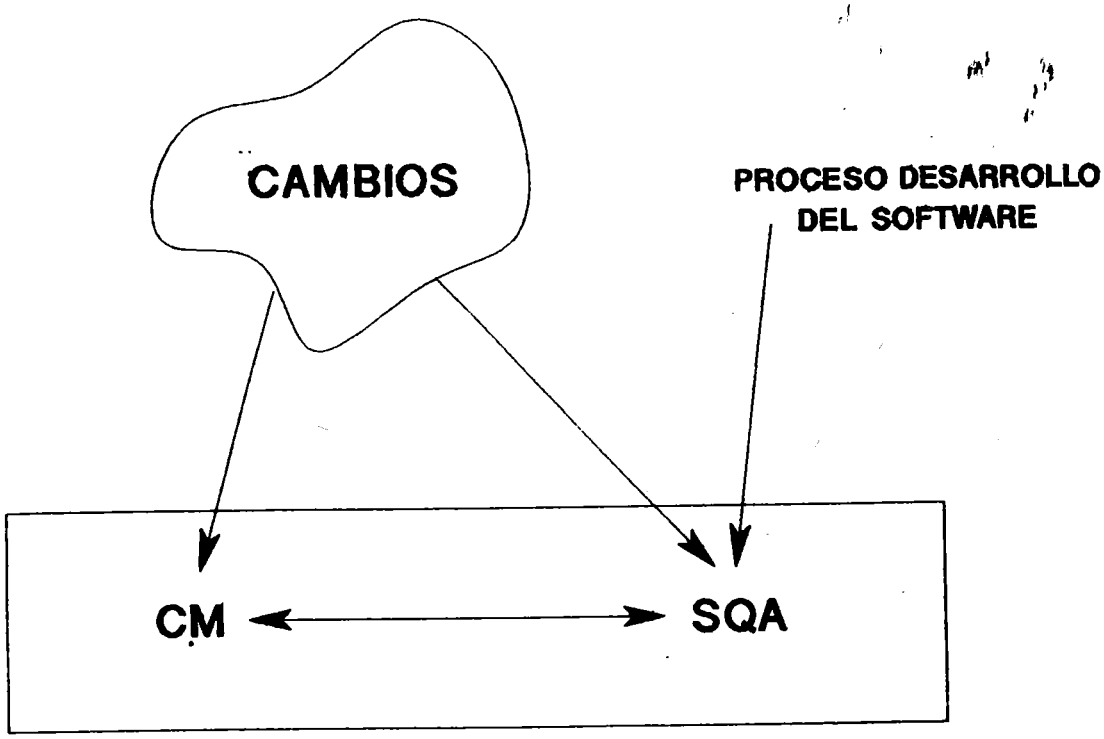
	MUY BAJO	BAJO	NORMAL	ALTO	MUY ALTO
RELY	1,35	1,15	1	0,98	1,1

error del libro de Boehm, propagado por el Sommerville

TAMAÑO PRODUCTO EN KDSI	MUY BAJO	BAJO	NORMAL	ALTO	MUY ALTO
2	1,25	1,12	1	0,9	0,81
8	1,3	1,14	1	0,88	0,77
32	1,35	1,16	1	0,86	0,74
128	1,4	1,18	1	0,85	0,72
512	1,45	1,2	1	0,84	0,7

MODP

INTERESANTE : EL COSTE DEL MTMO SE DIVIDE POR DOS (PARA UN SW DE 128 KDSI) SI SE EMPLEAN INTENSIVAMENTE MODP.



CM: CONFIGURATION MANAGEMENT

SQA: SOFTWARE QUALITY ASSURANCE

CALIDAD: TIENE QUE SER CONSTRUIDA EN EL PRODUCTO SOFTWARE A TODO LO LARGO DEL PROCESO. NO SE PUEDE "AÑADIR" EN LAS ETAPAS DE DEPURACION Y PRUEBA (GARY FORD, 1990)

* ¿A QUE CLIENTES SE LES HA ENTREGADO UNA DETERMINADA VERSION DEL SISTEMA?

* ¿QUE CONFIGURACION HW/SW (S.O.) SE REQUIERE PARA EJECUTAR UNA DETERMINADA VERSION DEL SISTEMA?

* ¿CUANTAS VERSIONES DE UN SISTEMA SE HAN CREADO Y EN QUE FECHAS?

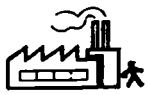
* ¿QUE VERSIONES DE UN SISTEMA SE VERAN AFECTADAS POR CAMBIOS EN TAL COMPONENTE CONCRETO?

* ¿CUANTAS PETICIONES DE CAMBIO SE HAN FORMULADO EN RELACION CON UNA DETERMINADA VERSION?

* ¿CUANTOS ERRORES ADVERTIDOS EXISTEN PARA UNA VERSION CONCRETA?

* ETC.

GESTION DE CONFIGURACIONES



RESPONSABILIDAD:

* SQA ES UNA FUNCION DE GESTION

* SUELE HACERSE POR UN EQUIPO DE PERSONAS AJENAS AL PROYECTO Y QUE INFORMA POR ENCIMA DE SU DIRECTOR: FUENTE DE CONFLICTOS

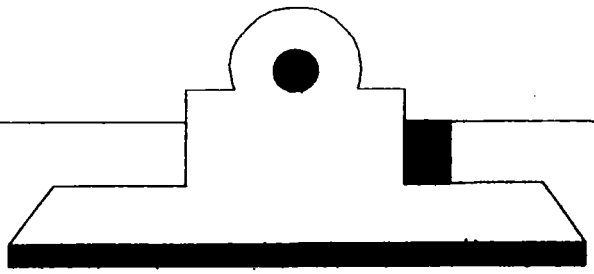
PROBLEMAS:

* LA NATURALEZA ELUSIVA DE LA CALIDAD DEL SW (RECORDAR CRITERIOS DE CALIDAD, BUCKLEY 1984)

* NO HAY CONSENSO EN CUANTO A LO QUE SIGNIFICA "ALTA CALIDAD" DEL SOFTWARE

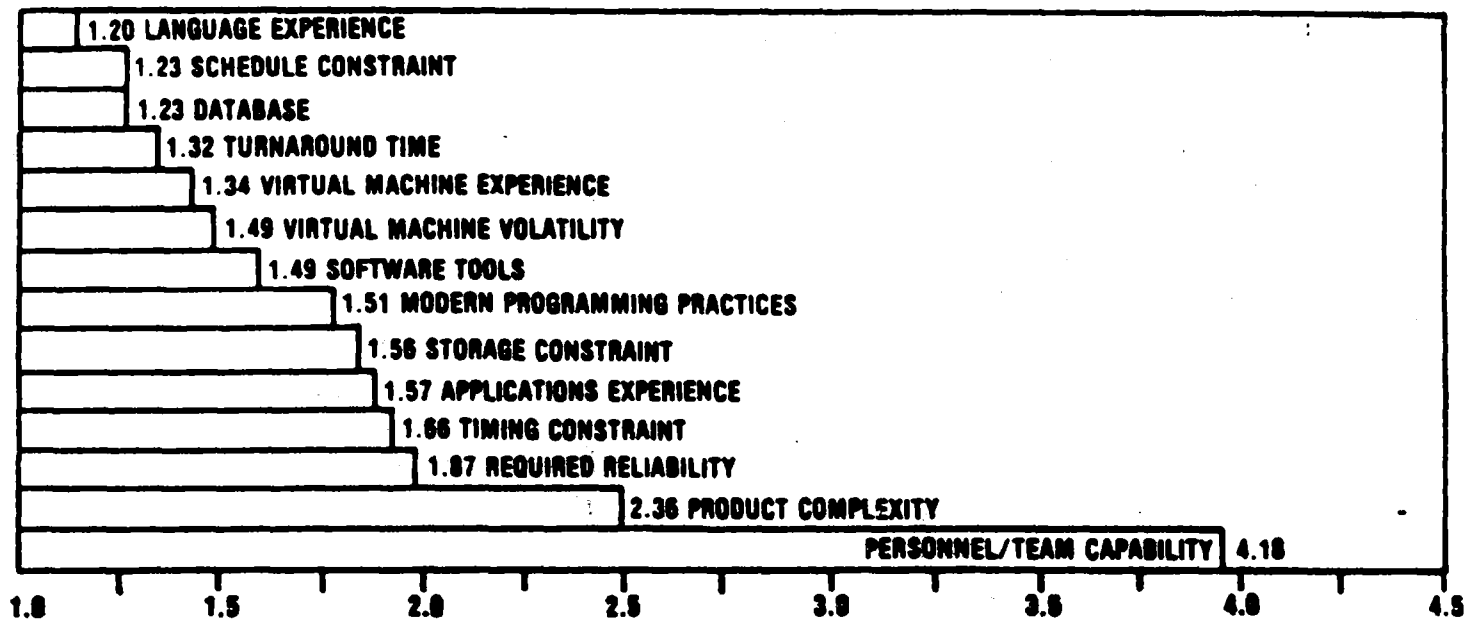
* RARA VEZ SE DISPONE DE METRICAS ADECUADAS

CONTROL DE CALIDAD



FACTORES HUMANOS INDIVIDUALES

- ☆ LIDERAZGO PROYECTO SOFTWARE
- ☆ ESTRUCTURA FACTORIAL DE LA INTELIGENCIA
- ☆ ESTRUCTURA FACTORIAL DE LA PERSONALIDAD
- ☆ ALGO DE PSICOLOGIA COGNITIVA. APLICACION EN PROGRAMACION
- ☆ FORMACION EN INGENIERIA DEL SOFTWARE


1.20 LANGUAGE EXPERIENCE
1.23 SCHEDULE CONSTRAINT
1.23 DATABASE
1.32 TURNAROUND TIME
1.34 VIRTUAL MACHINE EXPERIENCE
1.49 VIRTUAL MACHINE VOLATILITY
1.49 SOFTWARE TOOLS
1.51 MODERN PROGRAMMING PRACTICES
1.56 STORAGE CONSTRAINT
1.57 APPLICATIONS EXPERIENCE
1.66 TIMING CONSTRAINT
1.87 REQUIRED RELIABILITY
2.36 PRODUCT COMPLEXITY
PERSONNEL/TEAM CAPABILITY
4.18
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

INDIVIDUO

LIDERAZGO TECNICO

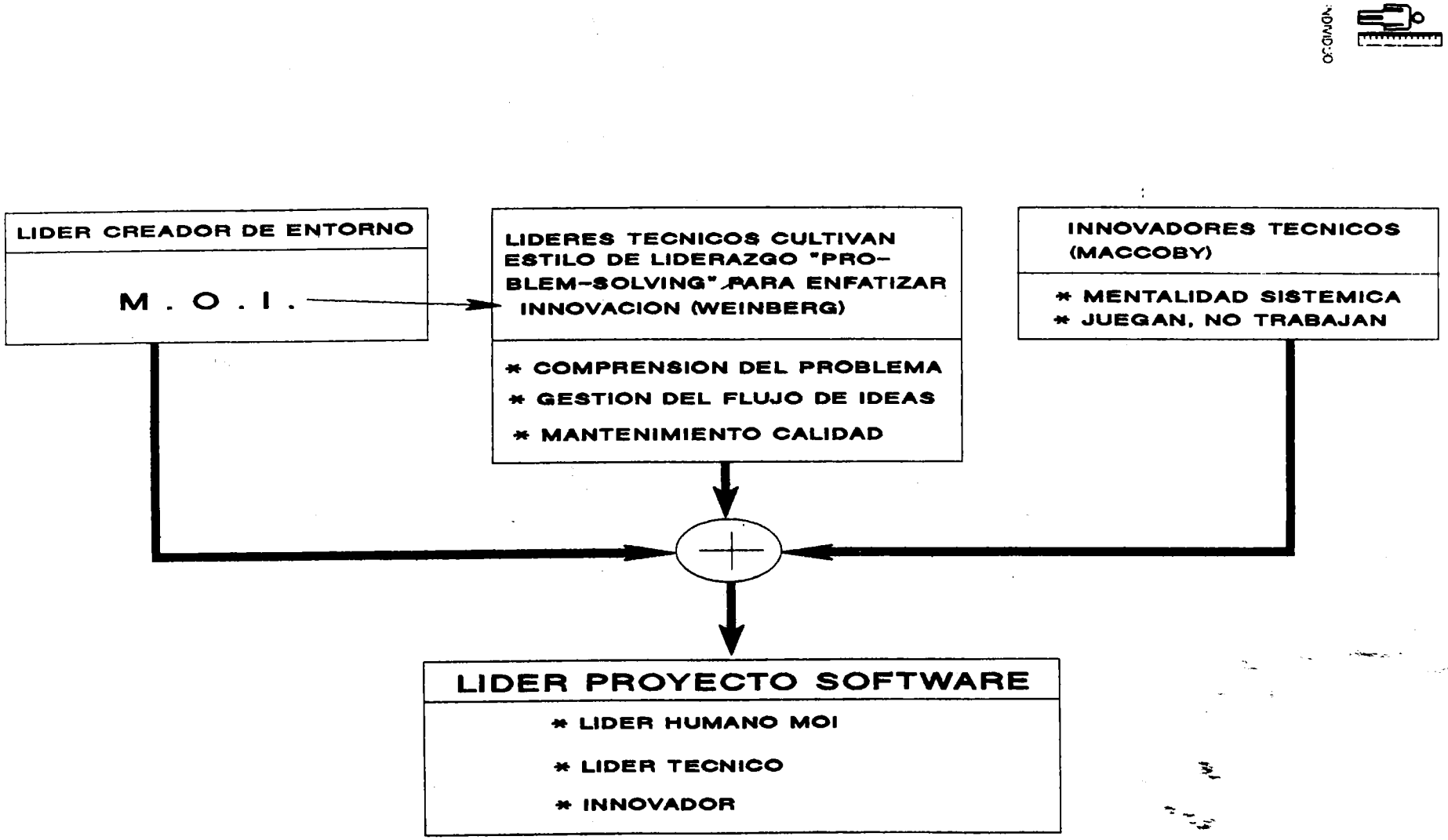
LIDERAZGO: ES EL PROCESO DE CREAR UN ENTORNO EN EL QUE LAS PERSONAS SE VEAN POTENCIADAS

MODELO ORGANICO DE LIDERAZGO M.O.I: LOS TRES INGREDIENTES QUE TIENE QUE POSEER EL ENTORNO

M: MOTIVACION

O: ORGANIZACION

I: IDEAS O INNOVACION ← **ENFASIS DEL LIDER TECNICO (WEINBERG)**

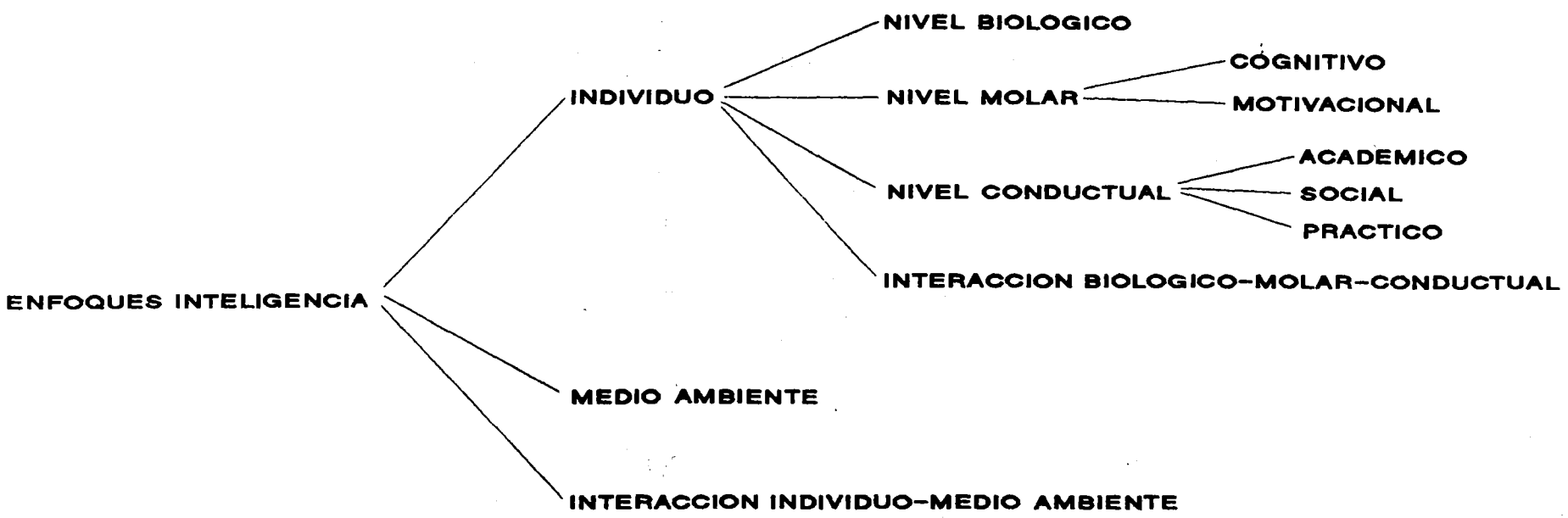

LIDER CREADOR DE ENTORNO
M . O . I .
LIDERES TECNICOS CULTIVAN ESTILO DE LIDERAZGO "PRO-BLEM-SOLVING" PARA ENFATIZAR INNOVACION (WEINBERG)
* COMPRENSION DEL PROBLEMA
* GESTION DEL FLUJO DE IDEAS
* MANTENIMIENTO CALIDAD
INNOVADORES TECNICOS (MACCOBY)
* MENTALIDAD SISTEMICA
* JUEGAN, NO TRABAJAN
LIDER PROYECTO SOFTWARE
* LIDER HUMANO MOI
* LIDER TECNICO
* INNOVADOR

INDIVIDUO

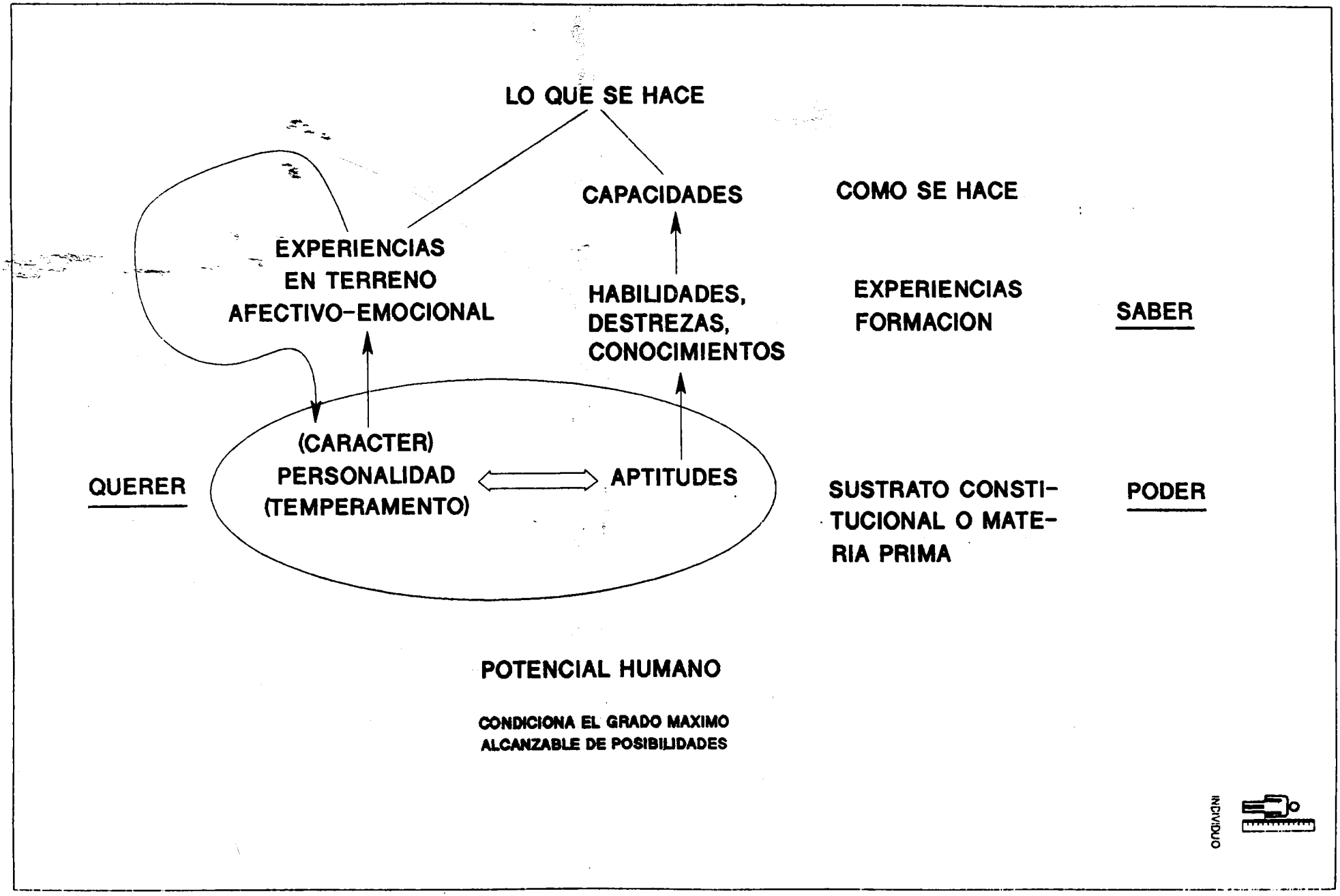
INNOVADORES TECNICOS EN EL DISEÑO DE SOFTWARE

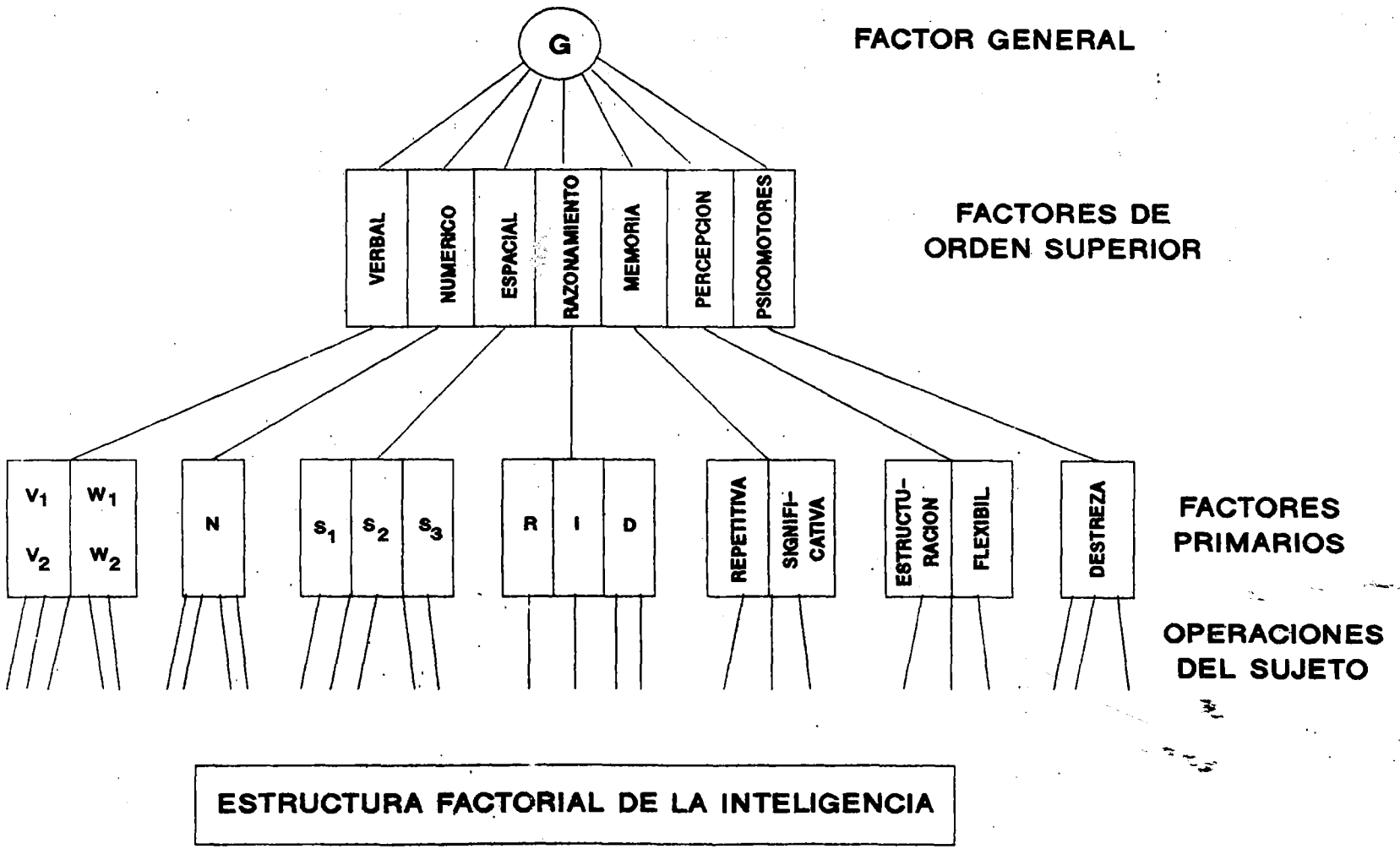
IDEAS, EXPERIENCIAS, METODOS Y PERSONALIDADES DE QUIENES HAN CREADO LA INDUSTRIA SOFTWARE

LEER: "PROGRAMADORES EN ACCION", DE SUSAN LAMMERS, MICROSOFT-ANAYA MULTIMEDIA, 1986



ESQUEMA DE CONCEPCIONES DE LA INTELIGENCIA (STERNBERG, 1986)


LO QUE SE HACE
CAPACIDADES
COMO SE HACE
EXPERIENCIAS
EN TERRENO
AFECTIVO-EMOCIONAL
HABILIDADES,
DESTREZAS,
CONOCIMIENTOS
EXPERIENCIAS
FORMACION
SABER
(CARACTER)
PERSONALIDAD
(TEMPERAMENTO)
APTITUDES
QUERER
SUSTRATO CONSTI-
TUCIONAL O MATE-
RIA PRIMA
PODER
POTENCIAL HUMANO
CONDICIONA EL GRADO MAXIMO
ALCANZABLE DE POSIBILIDADES
INDIVIDUO



ESTRUCTURA FACTORIAL DE LA INTELIGENCIA

INDIVIDUO

CAMPOS DE APTITUD

(Según J.L. Pinillos, en La Mente Humana)

El CAMPO VERBAL se refiere al uso inteligente del lenguaje y comprende otros tipos de aptitud: la **comprensión verbal (V)** y la **fluidez verbal (W).** Dentro de la comprensión verbal cabe distinguir todavía dos factores: el factor V_1, de carácter predominantemente lingüístico sintáctico, y el factor V_2 de carácter predominantemente semántico. El primero viene definido por preguntas que implican cierto dominio de la gramática, ortografía, corrección de frases, etc., y el segundo por la riqueza de vocabulario. Los factores de fluidez verbal se refiern a la facilidad para escribir muchas palabras que empiecen, por ejemplo, con una letra dada, la prontitud y abundancia de lenguaje y la riqueza y la facilidad para verbalizar ideas.

El CAMPO NUMERICO está representado por el factor N, definido por tests de rapidez y exactitud de cálculo numérico que, por otra parte, no presuponen necesariamente talento matemático de ninguna clase. Su contenido, mínimamente inteligente, guarda relación práctica con ciertos trabajos de tipo rutinario.

El CAMPO ESPACIAL abarca un conjunto de aptitudes precisas para resolver problemas de tipo técnico-práctico. Tales aptitudes son: el factor S_1 **espacial estático,** que capacita para resolver problemas espaciales en los que hay cambio de lugar y aspecto, pero no de estructura interna, como por ejemplo en los tests de rotación de figuras. El factor S_2 **espacial dinámico,** más importante para la inventiva técnica, que facilita la comprensión y manipulación imaginativa de complejos espaciales que cambian de estructura interna al desplazarse, como, por ejemplo, ocurre en los tests de desarrollo de superficies. El factor S_3 **espacial topológico,** que consiste en la aptitud parqa manipular mental o físicamente aspectos no figurativos ni métricos del espacio, como orientaciones, trayectoria, obstáculos, etc. Las profesiones técnicas suelen requerir cierta preeminencia de estos factores en los individuos que las esempeñan.

El CAMPO DE LA INTELIGENCIA FORMAL abarca un conjunto de factores que trascienden todo contenido y están presentes en la solución de cualquier tipo de problema, bajo la forma de reglas y operaciones lógicas. Entre los factores que parecen más claramente definidos se encuentran los de **razonamiento (R), deducción (D) e inducción (I).**

El factor R, probablemente una combinación de la deducción, la inducción y

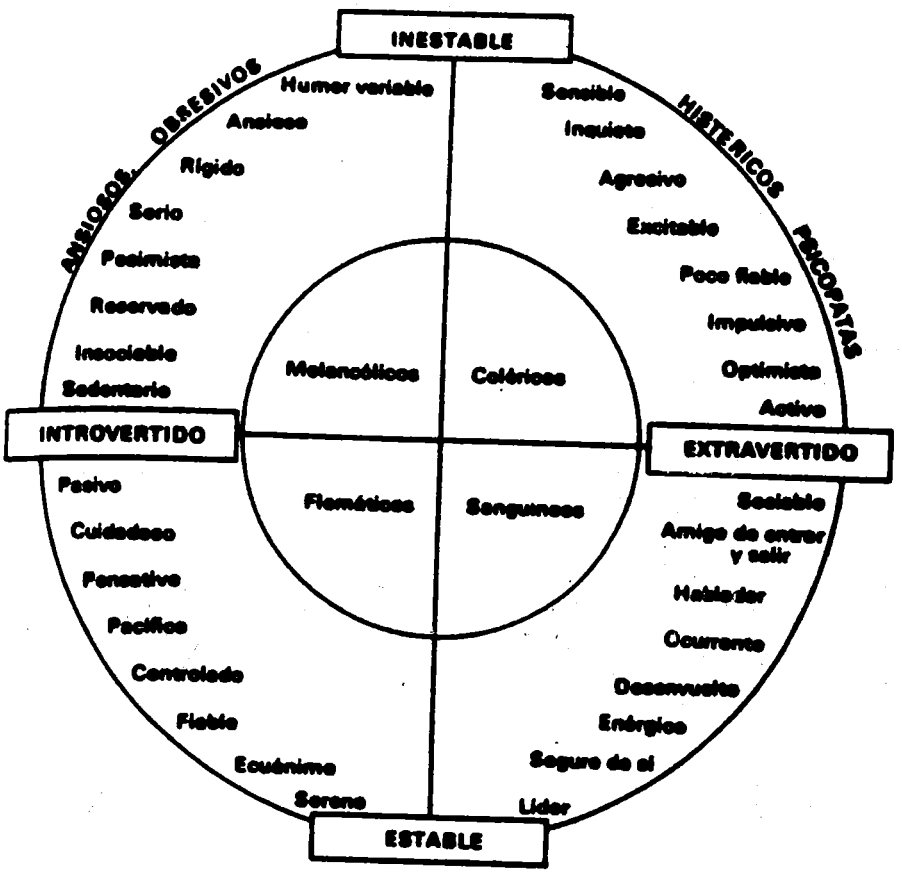
INDIVIDUO

otras operaciones menos conocidas, se refiere al razonamiento efectuado en condiciones restringidas, como exige la solución de problemas matemáticos. El factor D está definido por el tipo de discurso silogístico y el factor I, que señala la aptitud para descubrir reglas o principios, por los diversos tests de series, enlos que a partir de unos datos hay que inferir la regla o ley que los rige y dar una respuesta que continúe el orden de la serie.

El CAMPO DE LA MEMORIA engloba muchas y muy diferentes clases de memoria. Existen muy variadas memorias **repetitivas,** cuya relación entre sí y con otros factores de la mente es casi nula; y hay también una memoria **significativa,** más unitaria que reconstruye inteligentemente las experiencias pasadas. Posiblemente, este último es un aspecto de lo que hay de más general en la inteligencia humana.

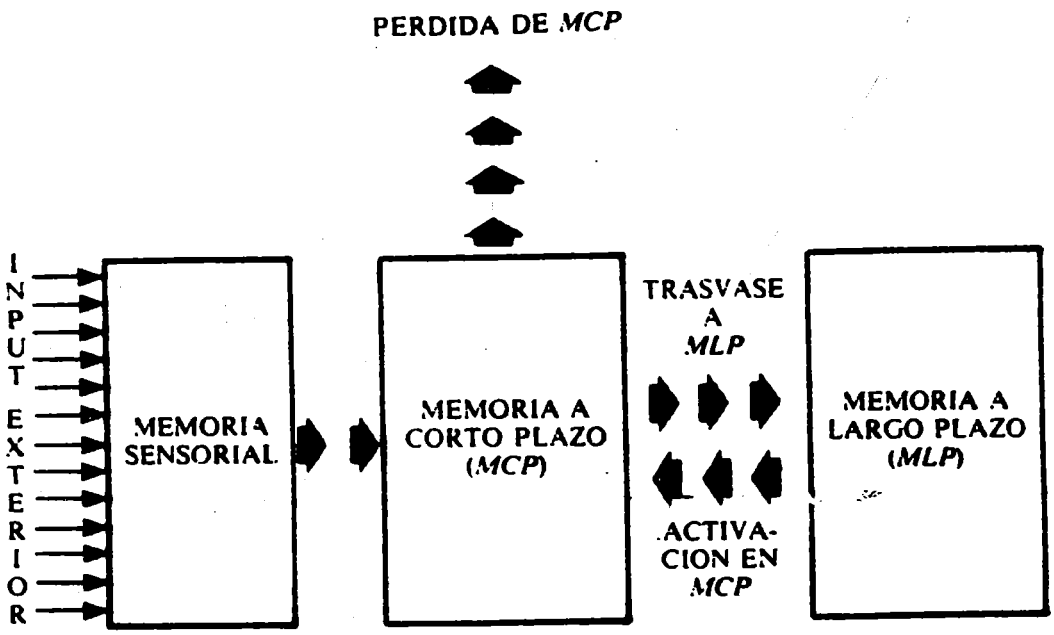
En el CAMPO PERCEPTIVO cabe distinguir dos tipos de factores: de una parte están los que se especifican por su materia o contenido (táctil, cromático, figural, etc) y, por otra, los de índole formal, caracterizados por el tipo de operación y no por el contenido de ésta. Entre estos últimos sobresalen los de **estructuración y flexibilidad** perceptivas. El primero de ellos consiste en la habilidad de ver totalidades de las que sólo se ofrece alguna parte; sus tests típicos son los de figuras mutiladas que el sujeto ha de completar mentalmente. El factor de flexibilidad perceptiva se refiere a la habilidad para romper mentalmente formas bien estructuradas y reestructurarlas de nuevo; los tests de dibujos camuflados camuflados son típicos de este factor. También poseen interés los factores de **rapidez y exactitud** perceptivas.

El CAMPO PSICOMOTOR está constituido por un verdadero enjambre de pequeños factores sin apenas correlación mutua. Entre los más importantes tests que miden este tipo de aptitudes se hallan los de **coordinación visuomotora, perseveración y ritmos.**



—Interpretación factorial, según Eysenck, de la estructura de la personalidad, en un sistema de dos dimensiones.

INDIVIDUO

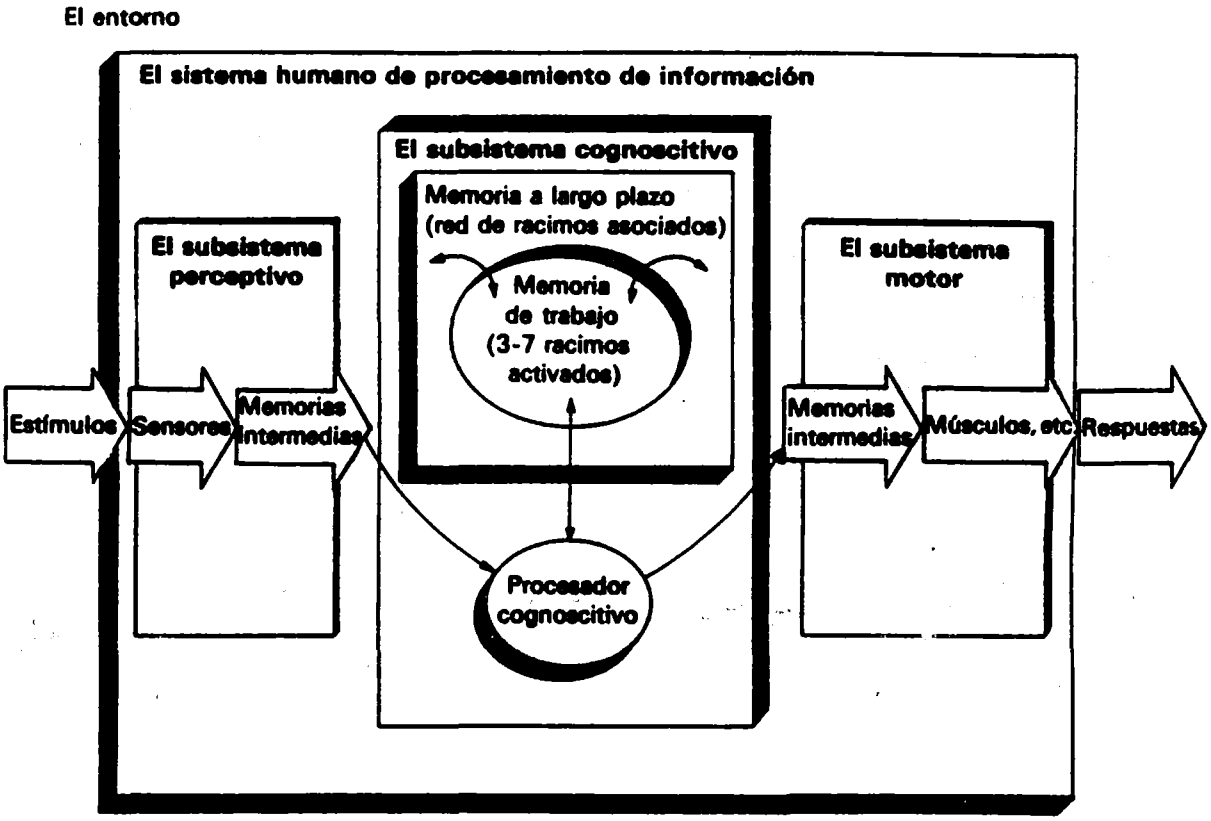


Modelo estructural de la memoria (basado en ATKINSON *y* SHIFFRIN, *1968).*

PSICOLOGIA COGNITIVA

INDIVIDUO

Esquema del sistema humano de procesamiento de la información



(HARMON, 1988)

INDIVIDUO

IMPLICACIONES PRACTICAS EN EL PROCESO DE DESARROLLO DE SOFTWARE

* Los programas no se comprenden sentencia por sentencia, sino que se abstraen informaciones en forma de rácimos que componen una estructura semántica interna. De ahí el interés de enseñar trozos de algoritmos elegantes, simples y potentes, que se memorizan en largo plazo. Después, el programador puede construir otras abstracciones utilizando estos elementos y puede producir versiones en los lenguajes que conozca.

* Las abstracciones de información son CONOCIMIENTO:

- Conocimiento **semántico**: operación de una sentencia de asignación, concepto de lista encadenada, concepto de operación del "hashing". Se almacena en formato independiente de su representación.

- Conocimiento **sintáctico**: detalles de representación sobre cómo escribir una declaración de procedure en Pascal, = o :=, etc.

* Problemas del programador novicio cuando está aprendiendo a programar:

a) Aprender los conceptos semánticos embebidos en el modelo computacional.

b) Aprender una sintáxis, a veces obscura y arbitraria.

c) Separar semántica y sintáxis

Este proceso no es bien comprendido por la mayoría de instructores. Un programador con experiencia, si cambia de lenguaje dentro de un mismo modelo computacional, tiene ya asumidas las fases a) y c).

*** Programación estructurada**

La estructura de la memoria de corto plazo se ve favorecida en su uso por el diseño descendente (sin necesidad de llamar información de otras partes del programa), por el rechazo del GOTO y por la repetición de dos o tres estructuras.

INDIVIDUO

*** Lenguajes y estructuras**

Los programas escritos en lenguajes cuyas sentencias codifican más directamente los conceptos semánticos de bajo nivel, como declaraciones de tipo, bucles WHILE, condicionales, etc. Mejor Pascal que Fortran, y éste emjor que ensamblador.

*** Facilidad de aprendizaje**

La habilidad de programar es independiente del lenguaje, por lo cual es relativamente fácil para un programador familiarizado con un lenguaje aprender otro del mismo tipo.

FORTRAN a Pascal (fácil)

FORTRAN a PROLOG (difícil, el modelo subyacente es muy diferente)

Programador novicio a PROLOG (más fácil que para un programador con experiencia; Programación funcional, lo mismo)

FORTRAN/Pascal a ADA (dificultad media, el modelo es el mismo, pero con constructos nuevos, como package, task, etc)

*** Organización formación programadores**

- Programadores con experiencia en el modelo: darles manual.

- Programadores sin experiencia: organizar estrategia, enfatizando la enseñanza global de las estructuras semánticas, primero.

* La ingeniería del software integra conocimiento semántico en tres áreas:

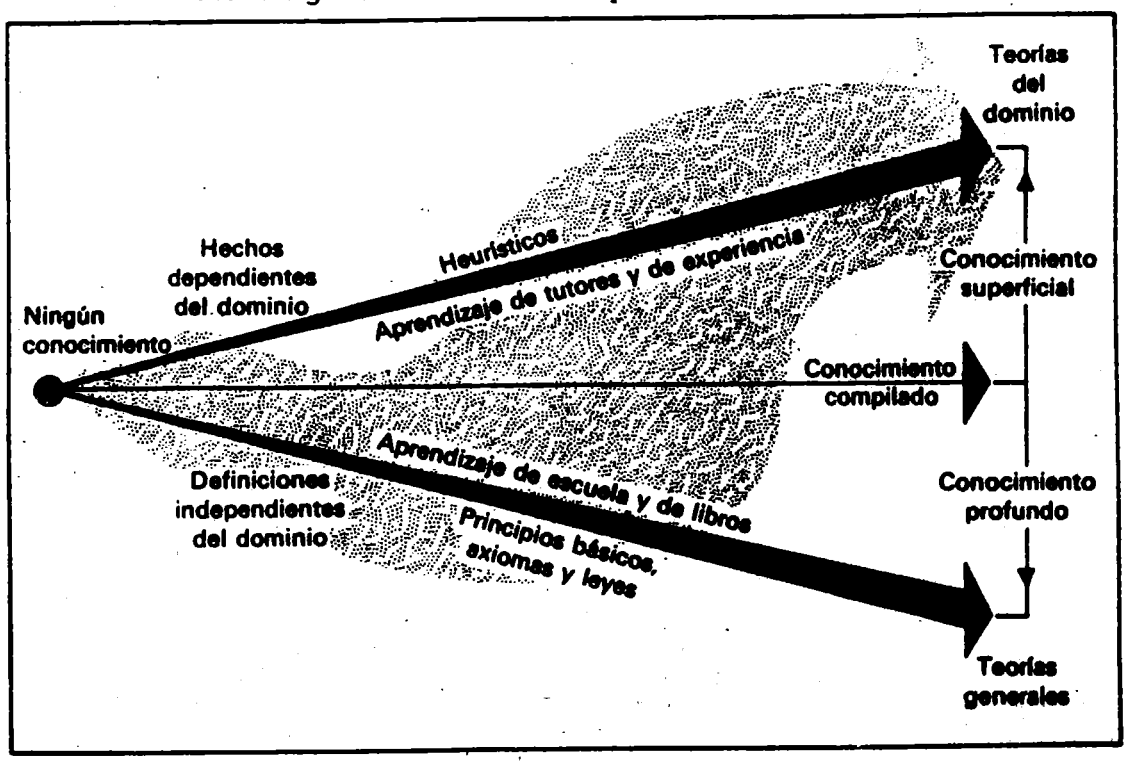
a) informática

b) tarea o problema implicado

c) gestión/organización

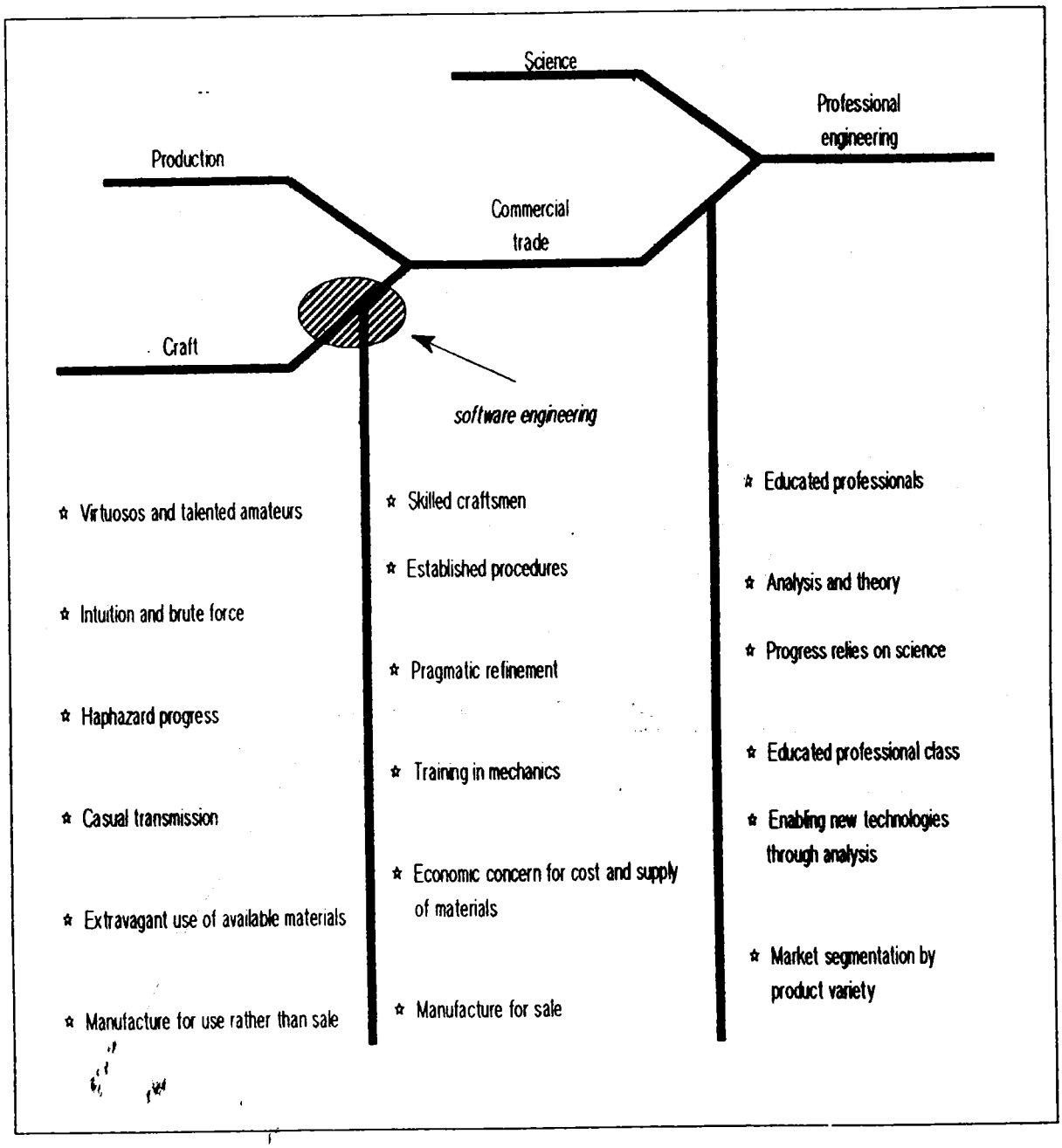
INDIVIDUO

Modelo general del desarrollo profesional



(HARMON, 1988)

INDIVIDUO



ESTADO DE LA INGENIERIA DEL SOFTWARE (SHAW, 1990)

INDIVIDUO

CUESTIONES SOBRE FORMACION EN INGENIERIA DEL SOFTWARE

Requirement	Semester Hours	Content Category
22.5%	27	Mathematics and Basic Sciences
37.5%	45	Software Engineering Sciences and Software Engineering Design
25%	30	Humanities, Social Sciences
15%	18	Electives

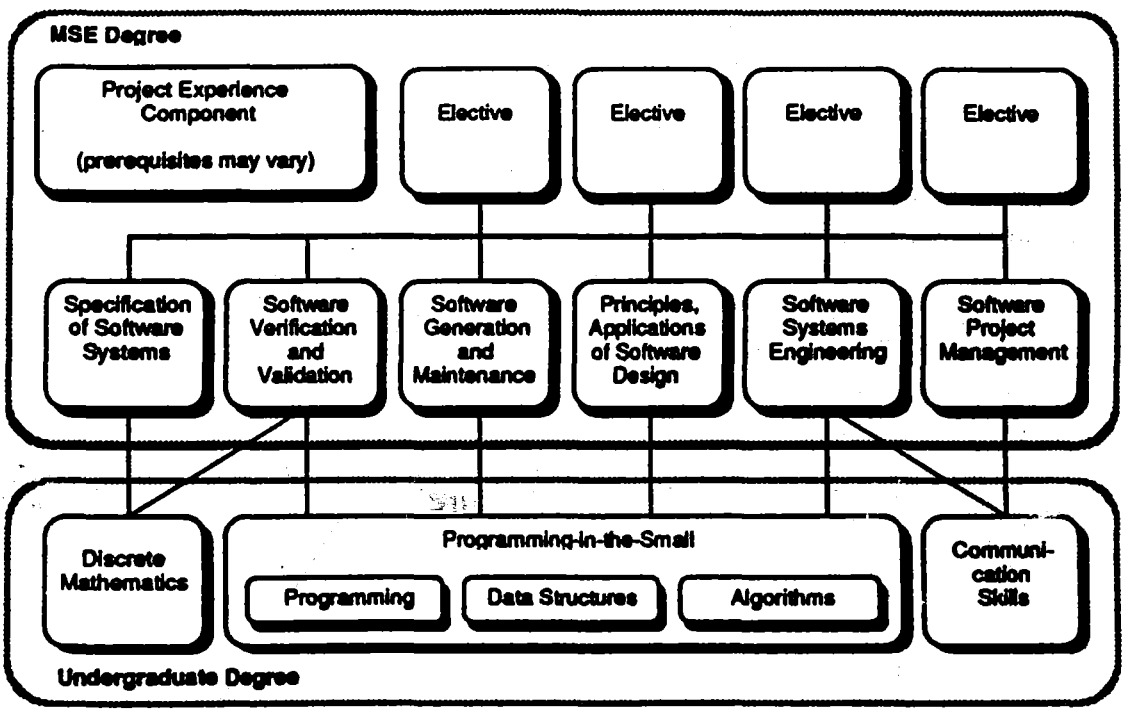
ESTRUCTURA DEL CURRICULO DE GRADO EN INGENIERIA DEL SOFTWARE U.S.A.

(EN GARY FORD, "1990 SEI REPORT ON UNDERGRADUATE SOFTWARE ENGINEERING EDUCATION", MARZO, 1990)

INDIVIDUO

MASTER EN INGENIERIA DEL SOFTWARE

(C.M.U.)



ESTRUCTURA DEL CURRICULO MSE

(EN M.ARDIS Y G.FORD, "1989 SEI REPORT ON GRADUATE SOFTWARE ENGINEERING EDUCATION, JUNIO 1989)

INDIVIDUO

3.4.6. Software Project Management

Catalog Description

This course addresses process considerations in software systems development. It provides advanced material in software project planning, mechanisms for monitoring and controlling projects, and leadership and team building.

Course Objectives

After completing this course, students should know how to develop a software project management plan; how to set up monitoring and controlling mechanisms for software projects; how to allocate and reallocate project resources; and how to track schedule, budget, quality, and productivity. In addition, students should understand the relationships among quality assurance, configuration management, and project documentation. They should gain an understanding of the key issues in motivating workers and leading project teams. They should be aware of intellectual property issues, software contracting and licensing, and process assessments.

Prerequisites

There are no specific prerequisites beyond admission to the MSE program.

Syllabus

Wks	Topics and Subtopics *(Objective)*

4 **Introduction** ***(Comprehension)***

Students need to see the "big picture" of software development. They also need to be motivated to study the problems of management.

- **Software engineering process**
  - **Process models (waterfall, incremental, spiral, rapid prototype, domain)**
  - **Organizational structures (functional, matrix, individual roles)**
- **Motivational case studies**
  - **Problematical projects (Project Foul, Medinet, *Scientific American*, OS/360, Multics, *Soul of a New Machine*)**
  - **Successful projects (GE RC2000, NASA space shuttle, ESS #1, Olympics message system)**
  - **Huge systems (air traffic control, Strategic Defense Initiative)**
- **Project origins**
  - **Requests for proposals (RFP), statements of work (SOW), contracts, business plans**
  - **System requirements**
  - **Software requirements**

INDIVIDUO

Legal issues
- Intellectual property rights
- Contracts
- Licensing
- Liability
- Post-employment agreements

4.5 Planning *(Application)*

Good planning is still considered an art rather than a science. However, students should learn how to use the best methods available. It is important to stress the importance of tailoring any method to the problem and the environment.

Standards
- External (2167A, 2168, NASA, IEEE)
- Internal (corporate, project)
- Tailoring

Work breakdown

Scheduling
- CPM, PERT, activity networks
- Milestones and work products

Resources
- Acquisition
- Allocation
- Tradeoffs

Risk analysis
- Identification
- Assessment
- Contingency planning

Estimates
- Expert judgment (individual, Delphi)
- Size estimates
- Models (driven by lines of code, by function point; time-sensitive models)

4.5 Monitoring and Controlling *(Application)*

Much of this topic deals with issues of product quality. There is an overlap here with material from the Software Verification and Validation course. The subtopic on leadership may be difficult to teach, but its inclusion in the course is important, if only to stimulate awareness of the different kinds of problems found in this area.

Process metrics
- Quality
- Schedule
- Budget
- Productivity

Earned value tracking

Quality assurance

INDIVIDUO

Technical reviews (walkthroughs, inspections, acceptance testing)
Planning

Configuration management
Planning
Identification
Change control
Auditing
Tools

Risk management
Tracking
Crisis management

Leadership, training, and motivation
Work environment
Motivation and job satisfaction
Leadership styles
Team structures (hierarchical, chief programmer, democratic)
Productivity assessment
Performance reviews
Small group dynamics

1 Project Assessment *(Application)*

Students should assess one another's work. This is one of the best ways to synthesize material from several topics of the course. For example, the combined effects of poor planning and poor control are best seen through postmortem analysis. Students should be given the opportunity to fail, since they will be less willing to try novel approaches outside academia.

In-process assessment

Final assessment

Project formation
Postmortems and lessons learned
Summary data collection
Staff reassignments

Relevant SEI Curriculum Modules

CM-3	*The Software Technical Review Process*, James S. Collofello
CM-4	*Software Configuration Management*, James E. Tomayko
CM-7	*Assurance of Software Quality*, Bradley J. Brown
CM-10	*Models of Software Evolution: Life Cycle and Process*, Walt Scacchi
CM-14	*Intellectual Property Protection for Software*, Pamela Samuelson and Kevin Deasy
CM-12	*Software Metrics*, Everald E. Mills
CM-21	*Software Project Management*, James E. Tomayko and Harvey K. Hallman

INDIVIDUO

Pedagogical Concerns

A project should be assigned; it should primarily involve planning–no implementation need be done.

It is difficult to provide motivation for many of the topics in this course without experience managing software development projects. Guest lecturers may be especially helpful for this.

Many aspects of software maintenance may be considered project management issues. Instructors should coordinate the coverage of these topics between this course and the Software Generation and Maintenance course.

Bibliography

The bibliography for this course is still being developed. The bibliography of curriculum module CM-21 provides useful references for most of this course.